Web Taiko band

Por

Alejandro Pedrosa García
Leila Ruiz Casanova
María Victoria Barylak Alcaraz

UNIVERSIDAD COMPLUTENSE
MADRID

Grado en Ingeniería Informática
Grado en Ingeniería del Software
Facultad de Informática

Dirigido por
Adrián Riesco Rodríguez

Madrid, Julio 2020

Sobre TEFLON

Teflon(cc0 1.0(documentación) MIT(código))es una plantilla de LaTeX creada por David Pacios Izquierdo con fecha de Enero de 2018. Con atribuciones de uso CC0.

Esta plantilla fue desarrollada para facilitar la creación de documentación profesional para Trabajos de Fin de Grado o Trabajos de Fin de Máster. La versión usada es la 1.3.

V:1.3 Overleaf V2 with pdfLaTeX, margin 1in, NO-bib

Contacto
Autor: David Pacios Izquiero
Correo: dpacios@ucm.es
ASCII: asciifdi@gmail.com
Despacho 110 - Facultad de Informática

Índice general

Resumen

Los juegos basados en el seguimiento de ritmos musicales han tenido un gran impacto en la industria del entretenimiento. La clave de su éxito ha sido la combinación de melodías con el desafío a la destreza de los jugadores a la hora de tocar instrumentos. Uno de sus mayores exponentes es el juego arcade *Taiko no Tatsujin* lanzado hasta en doce plataformas distintas. Este juego se basa en seguir un ritmo musical establecido acertando las notas sobre un tambor o *"taiko"*.

En este proyecto se propone el desarrollo web de un juego de ritmos musicales basado en este famoso pasatiempo, pues *Taiko no Tatsujin* todavía no ha sido sacado al mercado como aplicación web. El desarrollo del juego pretende ser un *fangame* dándole importancia al modo multijugador y ofreciendo a los usuarios la posibilidad de crear sus propios niveles. La implementación de este juego se hará empleando *JavaScript* como lenguaje principal de la aplicación junto con tecnologías web modernas como es *Node.js*.

Palabras clave: ritmos musicales, *Taiko no Tatsujin*, aplicación web, *fangame*, multijugador, *JavaScript*, *Node.js*

Abstract

Games based on the following up of musical rhythms have had a great impact in the entertainment industry. The key to their success has been the combination of melodies with the challenge of the players' skills when playing different instruments. One of its main exponents is the arcade game *Taiko no Tatsujin* released for up to twelve different platforms. This game is based on following an established musical rhythm hitting the notes on a drum or *"taiko"*.

In this project the web development of a musical rhythm game based on this famous pastime is presented, due to the fact that *Taiko no Tatsujing* has not been released as a web application yet. The development of this game is expected to be a fangame giving importance to the multiplayer mode and offering users the possibility of creating their own levels. The implementation of this game will be done using *JavaScript* as the main application language in conjunction with modern web technologies like *Node.js*.

Keywords: musical rhythms, *Taiko no Tatsujin*, web application, fangame, multiplayer, *JavaScript*, *Node.js*

Capítulo 1

Introducción

En 2001 una famosa marca de videojuegos lanzó al mercado un juego de arcade que, debido a su éxito, relanzaría en hasta doce plataformas diferentes a lo largo de las últimas dos décadas. Este videojuego no es otro que *"Taiko no Tatsujin"*, un juego de categoría musical que consiste en conseguir acertar las notas de la canción seleccionada mediante un mando especial con forma de *"taiko"* o tambor. El juego presenta unos círculos de colores en la pantalla, desplazándose horizontalmente por ésta hasta una posición marcada en la que golpear el *"taiko"*.

Figura 1.1: Máquina arcade de *Taiko no Tatsujin* (fuente: Amazon)

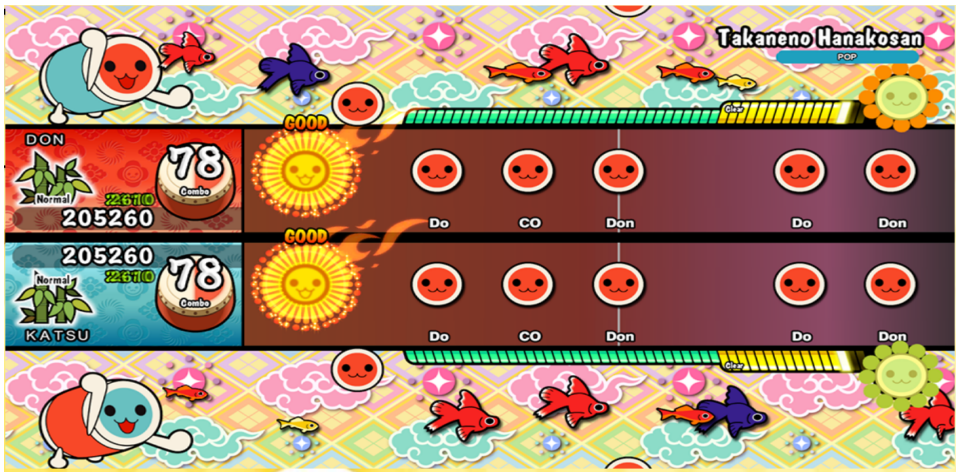

Figura 1.2: Versión *online* de *Taiko no Tatsujin* (fuente: página oficial de Bandai Namco)

En la figura 1.1 se puede ver una máquina recreativa del juego de arcade, mientras que la figura 1.2 representa la versión *online* del juego que ha servido como inspiración para este proyecto.

1.1. Peko Peko no Neko

Pekopeko no Neko es un juego que toma como base la versión para ordenador de este videojuego, partiendo de cero y cambiando por completo el diseño artístico que suele caracterizar a este famoso pasatiempo.

Peko Peko no Neko podría traducirse como "gato hambriento" y es en este concepto en el que se ha basado el diseño del proyecto. Mientras que *"Taiko no Tatsujin"* representa las notas de las canciones mediante círculos de distintos colores y tamaños, Peko Peko no Neko las representa mediante famosas comidas japonesas, como se puede ver en la figura 1.3. Siguiendo este concepto, la posición de meta en la cual se ha de acertar las notas está representada por un plato.

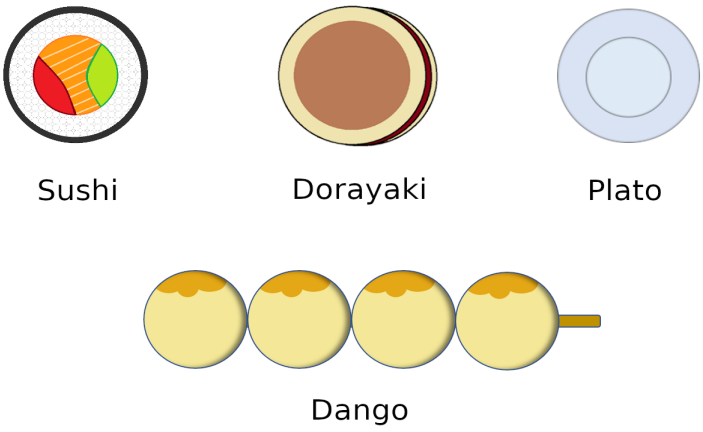

Figura 1.3: Tipos de círculos

El objetivo de Peko Peko no Neko es el mismo que el del juego en la versión web [1], intentar acertar las notas de la canción seleccionada cuando pasen por la posición de meta. Los círculos representados mediante *sushis* o *dorayakis* equivalen a una nota de la canción y, por tanto, a una pulsación del usuario. Sin embargo, los círculos representados por los *dangos* equivalen a varias pulsaciones.

Además de este objetivo, nuestra versión permite a los usuarios la posibilidad de crear niveles y le da importancia al juego en modo multijugador, permitiendo que dos personas jueguen al mismo tiempo y desde el mismo terminal. Cabe destacar que este proyecto es una versión del juego original (*fangame*) y en ningún momento pretende ser una copia del mismo.

Se decidió implementar esta versión del juego como Trabajo de Fin de Grado debido a la magnitud del reto que suponía su desarrollo. Ninguno de los miembros del equipo había desarrollado en su etapa universitaria un proyecto de esta envergadura, no se habían utilizado las tecnologías con las que se ha desarrollado de esta manera, y parecía una forma de asentar el aprendizaje de algunas de las tecnologías que ya se conocían y de aprender otras nuevas que aún no se habían utilizado o no se habían llevado a la práctica en un supuesto más allá de las entregas propuestas por la universidad.

1.2. Objetivos

El objetivo principal de este proyecto ha sido el desarrollo de una versión web del juego de arcade *"Taiko no Tatsujin"*, añadiendo características adicionales como puede ser el juego en modo multijugador o la posibilidad de crear niveles. Para lograr dicho objetivo se plantearon los siguientes objetivos:

- Obtener un mayor entendimiento de cómo se implementan los juegos basados en ritmos musicales.
- Investigar distintos *frameworks* de desarrollo web.
- Afianzar nuestros conocimientos tanto de tecnologías de desarrollo web como de bases de datos.
- Diseño de GIFs e imágenes para la parte visual del juego.
- Implementar una opción de modo multijugador al juego.
- Permitir a los usuarios la creación de niveles de juego propios.

1.3. Plan de trabajo

El plan de trabajo seguido para la realización de este proyecto se puede dividir en cinco fases principales que se corresponden con los distintos objetivos planteados (véase sección 1.2). Cabe destacar que se ha llevado a cabo un proceso de investigación simultáneo a la implementación en las fases iniciales del proyecto.

En primer lugar se llevó a cabo una investigación sobre el funcionamiento de la versión web de *Taiko no Tatsujin*. Esta investigación serviría para formar una idea sobre la estructura que tendría el proyecto. Además, se investigaron también distintos procedimientos a seguir a la hora de dibujar los círculos en la pantalla para conseguir un efecto de desplazamiento.

Más tarde se llevó a cabo una investigación sobre los posibles *frameworks* de desarrollo web y motores de bases de datos con los que implementar el proyecto.

Una vez elegido tanto el *framework* como la base de datos se procedió a la implementación por módulos del proyecto, de forma que se contara con una versión funcional del juego para un único jugador. A la par que se desarrollaba dicha versión se avanzaba con el modo multijugador, de forma que no existiensen grandes diferencias en el desarrollo de cada modo de juego.

Por último, una vez se terminó la versión funcional del juego, que contaba con dos modos de juego, individual y multijugador, se procedió al desarrollo de la funcionalidad de crear nivel, de forma que se permitiese a los usuarios la creación de sus propios niveles de juego.

1.4. Estructura de la memoria

A partir de este punto, la memoria va a seguir una estructura dividida por capítulos, siendo estos los siguientes:

- Capítulo 3: Tecnologías y metodología, donde se mencionarán las distintas tecnologías utilizadas para desarrollar el proyecto, así como la metodología de desarrollo que se siguió.
- Capítulo 4: Arquitectura e implementación, donde se explicará la estructura y el funcionamiento del proyecto a nivel técnico.
- Capítulo 5: Bases de datos, en el cuál se explicará la estructura de los datos almacenados, así como las funciones implementadas para su manipulación.
- Capítulo 6: *Running tool*, donde se mostrarán todas las posibles interacciones del usuario con la aplicación.
- Capítulo 7: Conclusiones y trabajo futuro, donde se exponen las coclusiones del proyecto, así como ideas para implementaciones futuras.
- Capítulo 9: Contribuciones al proyecto, donde se explicará qué ha hecho cada uno de los integrantes del grupo.

1.5. Repositorio

El código del juego está disponible en el siguiente repositorio de *GitHub*: https://github.com/Monosa/PekoPekoNoNeko

Capítulo 2

Introduction

In 2001 a popular video game brand launched on the market an arcade game that, due to its success, would be relaunched in up to twelve different platforms over the past two decades. This video game is *"Taiko no Tatsujin"*, a music category video game in which players have to hit the notes of the selected song using a *"taiko"* or drum controller. The game presents a series of colored circles that appear on the screen, moving horizontally across it to a marked position in which the player must hit the *"taiko"*.

Figura 2.1: *Taiko no Tatsujin* arcade machine (source: Amazon)

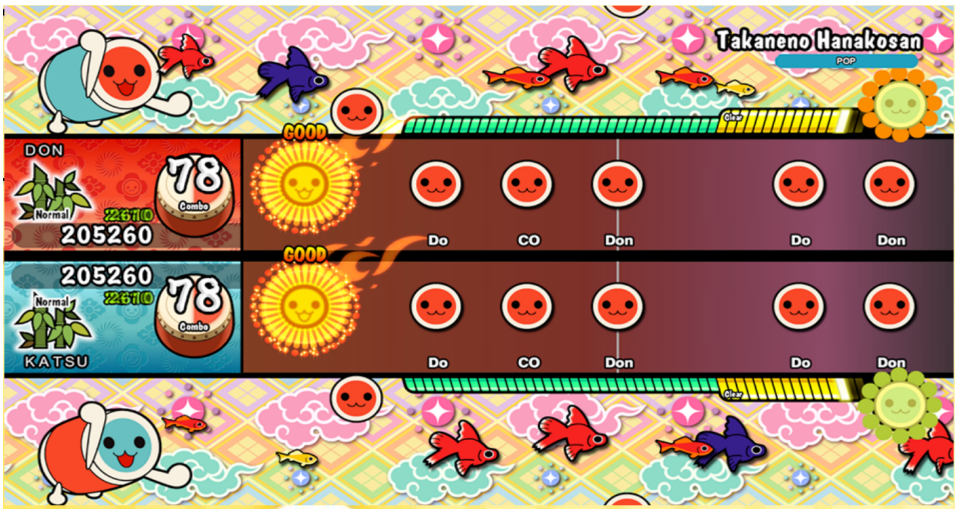

Figura 2.2: Online version of *Taiko no Tatsujin* (source: official website of Nintendo)

In the figure 2.1 we can see an arcade game machine, while the figure 2.2 represents the online version of the game which has been our inspiration to develop this project.

2.1. Peko Peko no Neko

Peko Peko no Neko is a game based on this video game's computer version, starting the developing from scratch and changing the artistic design that usually characterizes this popular game.

Peko Peko no Neko could be translated as "hungry cat" and this is the concept which has determined the artistic design of the project. While *"Taiko no Tatsujin"* displays the song notes using circles of different colors and sizes, Peko Peko no Neko represents them using popular Japanese dishes, such as those shown in the figure 2.3. Following this concept, the target position in which we must hit the notes is represented by a plate.

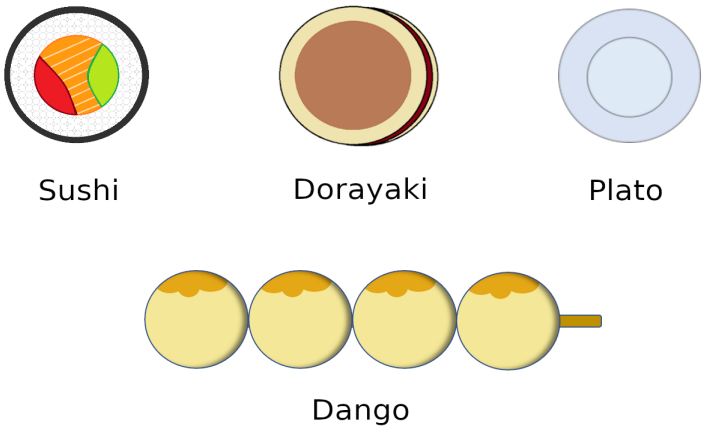

Figura 2.3: Circle types

The goal of Peko Peko no Neko is the same as the one in the web version of game [1], try to hit the notes of the selected song when these notes pass over the target position. The circles represented with *sushis* and *dorayakis* are equivalent to one sole song note and therefore, a players' single keystroke. However, the circles represented with *dangos* are equivalent to several keystrokes.

In addition to this objective, our version gives the user the possibility to create their own levels and grants importance to the multiplayer mode, letting two players play at the same time and in the same terminal. It is worth mentioning that this project is a fan version of the original game and it is not intended to be a copy of it.

We decided to implement this version of the game as our final year dissertation due to the magnitude of the challenge its development entailed. None of the team members had developed a project of these proportions during their university degree, some of the technologies used were unknown to the team, and it seemed a good way to settle our knowledge of the technologies which had been used and to learn others that we hadn't had used yet.

2.2. Objectives

Our main goal in this project has been the development of a web version of the original arcade game *"Taiko no Tatsujin"*, adding new features such as the multiplayer mode or the possibility of creating new original levels. In order to achieve that goal, the following objectives were set:

- Getting a better understanding of how games based in musical rhythms are implemented.
- Investigating different web development frameworks.
- Strengthen our knowledge of web developing technologies and databases.
- Designing original GIFs and images for the artistic side of the game.
- Implementing a multiplayer mode to the game.
- Allow users to create original levels to play.

2.3. Work plan

The work plan followed for the development of this project can be divided in five different phases which correspond to the set objectives (see section 2.2). It is worth mentioning that an investigative process was carried out alongside the implementation process in the initial phases of the project.

In the first place the team carried out an investigation to determine the functioning of the web version of *Taiko no Tatsujin*. This investigation would help define the structure of the project. Also, the team investigated different methods of drawing the circles on the

screen to achieve a shifting effect.

Later another investigation was carried out to figure out which web development framework and data base engine would be used to implement the project.

Once the framework and the data base were chosen the team proceeded with the implementation of the different modules of the project, in order to achieve a functional version for a single player. At the same time the multiplayer mode was being implemented, so as to not have big differences in the development of each game mode.

Lastly, once the final version of the game was finished, which had two game modes, single and multiplayer, the team proceeded with the development of the functionality which allows users to create their own levels.

2.4. Report organization

From this point on, the report will follow a structure divided in chapters, which are the following:

- Chapter 3: Technologies and methodology, in which the different technologies utilized in the project are mentioned, just like the followed methodology.
- Chapter 4: Architecture and implementation, where the structure and the functioning of the project will be explain from a technical point of view.
- Chapter 5: Data base, in which the structure of the stored data is explained just like the functions implemented to manipulate it.
- Chapter 6: Running tool, where the different user interactions will be shown.
- Chapter 7: Conclusions and future work, where the project conclusions are exposed together with ideas for future implementations.
- Chapter 9: Project contributions, where each team member will explain what aspects of the project they undertook.

2.5. Repository

The code of the game is available in the following *GitHub* repository: https://github.com/Monosa/PekoPekoNoNeko

Capítulo 3

Tecnologías y metodología

En este capítulo se presentan las diferentes tecnologías utilizadas a lo largo del desarrollo del proyecto, así como la metodología empleada para facilitar y optimizar el trabajo cooperativo.

3.1. Tecnologías usadas

El entorno de desarrollo utilizado para la implementación de este trabajo fue *Visual Studio Code* un editor de código simplificado con soporte de operaciones de desarrollo como son la depuración de código o el control de versiones [2]. La razón de esta elección fue la sencillez de la interfaz gráfica, ya que solo provee de las herramientas necesarias para ciclos rápidos de programación, compilación y depuración. Además, este Entorno de Desarrollo Integrado (IDE) ofrece la posibilidad de instalar extensiones que facilitan la programación, como puede ser *Beautify*, que estructura el código de forma que sea más legible.

Para la base de datos, tal y como se explica en el capítulo 5, se utilizó *MongoDB*, una base de datos distribuida basada en documentos *JavaScript Object Notation* (JSON), cuyas consultas también se realizan en dicho formato, lo que facilita su programación e integración con el código [3]. Para facilitar la gestión de los datos almacenados, se utilizó *MongoDB Compass*, la interfaz gráfica de usuario de *MongoDB*. Esta herramienta permitió la inserción en la base de datos varios documentos a la vez mediante un fichero en formato JSON.

En cuanto tecnologías web, se optó por utilizar *Node.js*, un entorno de ejecución para *JavaScript* orientado a eventos asíncronos [4], junto con *Express.js*, una infraestructura de aplicaciones web mínima y flexible [5].

Para el sistema de control de versiones se utilizó *GitHub*, una plataforma de desarrollo colectiva en la que se puede alojar y revisar código, y gestionar proyectos software, basada en el sistema de control de versiones *Git* [6].

3.2. Metodología empleada

Para gestionar el trabajo a realizar de forma sencilla y evitar solapamientos entre los distintos miembros del grupo, se hizo uso de la herramienta de gestión de proyectos que ofrece *GitHub*. Esta herramienta permite plasmar de forma visual, mediante un tablero con distintas columnas, las diferentes tareas a realizar y el estado de las mismas.

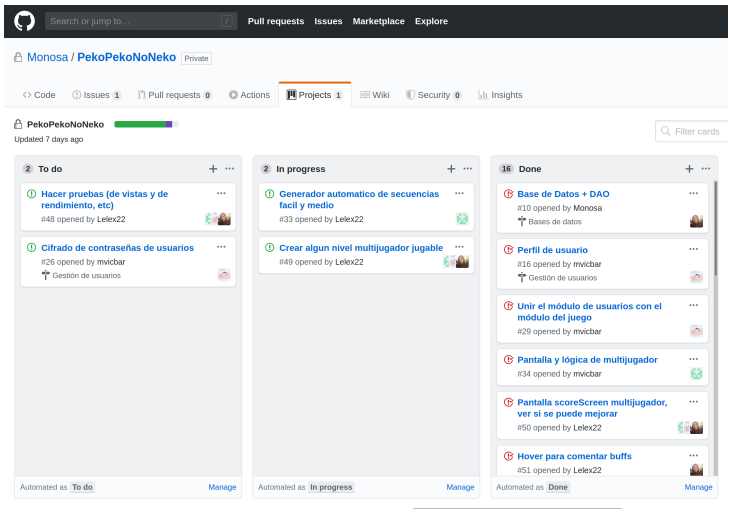

Figura 3.1: Tablero de *GitHub* para la gestión de las tareas

Como se puede ver en la figura 3.1 nuestro tablero contaba con tres columnas distintas. En la primera columna «To do», se colocaban las tareas aún por realizar. A continuación, en la columna «In progress», se indicaba qué tareas se estaban desarrollando y qué miembro del equipo las estaba implementando. Por último estaba la columna «Done», para aquellas tareas finalizadas.

Cada tarea se representaba mediante una tarjeta y equivalía a un *issue* de *GitHub*, que se cerraba una vez completada dicha tarea.

Para definir tanto las tareas como las funcionalidades a desarrollar se llevaban a cabo reuniones periódicas con nuestro tutor, más o menos cada 15 días. En estas reuniones se mostraban aquellas funcionalidades que habían sido implementadas hasta el momento, y se discutían las funcionalidades que deberían implementarse para la siguiente reunión. Una vez aclaradas dichas funcionalidades, se definían las tareas a realizar, las cuales se plasmarían en el tablero mencionado anteriormente.

Capítulo 4

Arquitectura e implementación

En este capítulo se describe en profundidad el proceso de diseño e implementación de Peko Peko no Neko, así como su arquitectura final. En la sección 4.1 se describe la primera aproximación al diseño del juego y en la sección 4.2 se presentan detalles técnicos de la implementación divididos por los principales módulos del juego.

4.1. Diseño del primer prototipo

En las primeras fases del proyecto nos centramos en la investigación del algoritmo base del juego, encargado de dibujar los círculos en la pantalla y comprobar la pulsación de las teclas por parte del usuario y la posición de cada círculo con respecto a la posición de meta.

Debido a esto, en los primeros meses de desarrollo implementamos un prototipo que consistía en un fichero *JavaScript*, llamado *main.js*, asociado a un fichero HTML estático, donde estaba incluido dicho algoritmo base.

El prototipo contaba con unos elementos básicos para conseguir una primera aproximación a la funcionalidad del juego. Los elementos implementados en este prototipo fueron: una pista donde se dibujaban los círculos, un cronómetro, un fichero de audio y un contador de puntos.

La pista donde se dibujan los círculos es un elemento <canvas> de HTML5. Decidimos utilizar este elemento ya que es la mejor forma de introducir gráficos y animaciones mediante el uso de *scripts* en una página web. El flujo principal de trabajo con un elemento <canvas> se basa en limpiar lo que esté dibujado en el propio <canvas>, guardar su estado en caso de que en todos los *frames* se quiera dibujar lo mismo, por ejemplo, el mismo fondo; dibujar las animaciones y restaurar el estado del <canvas> si se ha guardado previamente.

La funcionalidad del cronómetro era una función que aumentaba en una unidad un contador de milisegundos y actualizaba elementos de texto que representaban los minutos, los segundos y los milisegundos transcurridos. Esta función se llamaba cada milisegundo a través del método *setInterval* de *JavaScript*.

En un principio, el cronometro fue implementado debido a la necesidad de capturar el tiempo en milisegundos de las pulsaciones de los usuarios dentro del juego para compa-

rarlo con el tiempo de dibujado de los círculos. Sin embargo, tanto la función como el elemento que representaba el cronómetro se eliminaron debido a un cambio en el punto de vista a la hora de comprobar los aciertos del usuario. Dicho cambio se explicará con mayor detalle en la sección 4.2.5 de este mismo capítulo.

El fichero de audio estaba asociado al atributo src de un elemento <audio>. Solo contábamos con un fichero y, para su reproducción, se añadió el atributo autoplay al campo <audio>, de forma que cuando se terminara de cargar el fichero HTML éste empezaría a reproducirse de forma automática. En fases más avanzadas del proyecto se añadieron más ficheros, así como la posibilidad de elegir cuál se cargaría (véase sección 4.2.2).

Por último la funcionalidad del contador de puntos estaba basada en la comparación del tiempo de pulsación del usuario y el tiempo de dibujado del círculo. Dicha comparación aumentaba o disminuía la puntuación del usuaris según unos márgenes de error definidos. Finalmente, llegamos a la conclusión de que la mejor manera de comprobar los aciertos del usuario era mediante la comparación de la posición del círculo con la posición de meta. Es decir, aprovechando que contábamos con la posición del primer círculo en movimiento, comparábamos dicha posición con la posición de meta en el momento de la pulsación. Se profundizará en esta funcionalidad en la sección 4.2.5.

4.2. Implementación de Peko Peko no Neko

Tras la finalización del primer prototipo funcional, que englobaba el dibujado de los círculos mediante el fichero *JavaScript* y la captura de pulsaciones del usuario, comenzamos la implementación de la aplicación web.

La aplicación web fue implementada usando el *framework* de desarrollo *Node.js*, basado en *JavaScript*, aunque previamente se investigó la posibilidad de implementarla mediante *Django*, un *framework* web en *Python* (véase capítulo 3). En la sección 4.2.1 se exponen los motivos que marcaron el cambio entre estos dos frameworks, así como la implementación definitiva en *Node.js*.

La aplicación está dividida en módulos que se ven representados dentro del proyecto como carpetas. Esta división se puede observar en la figura 4.1. A continuación, vamos a explicar las funcionalidades que ofrece cada módulo, así como su proceso de implementación.

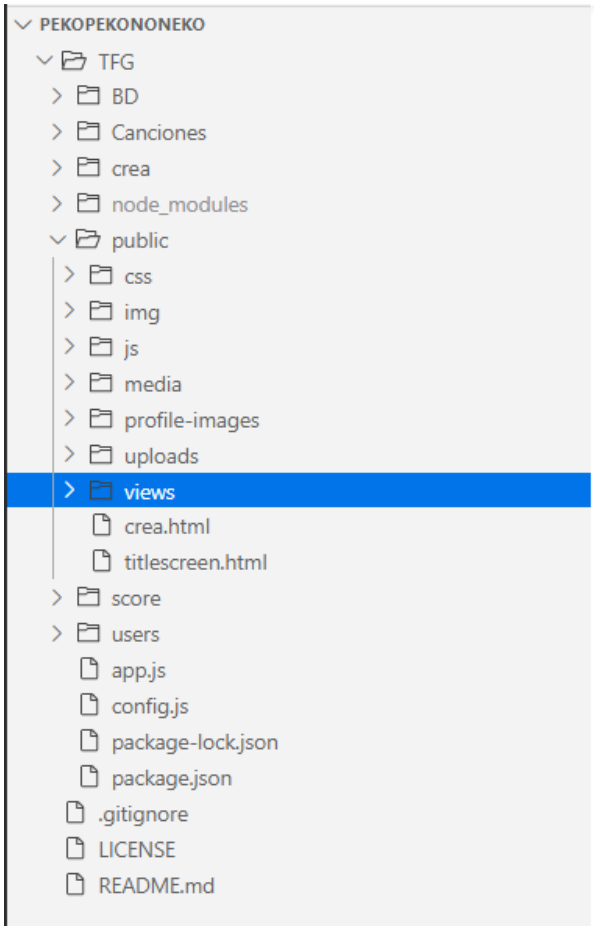

Figura 4.1: Estructura en carpetas del proyecto

4.2.1. Módulo de usuarios

El desarrollo se inició con el módulo de usuarios, encargado de gestionar el registro, el acceso y las modificaciones de los datos de los usuarios de la aplicación.

La idea principal era desarrollar una gestión básica de usuarios que permitiese a los mismos registrarse en la aplicación, así como acceder a ella a través de sus credenciales e integrar el prototipo del juego que previamente se había implementado.

Como se ha mencionado anteriormente, la primera aproximación se llevó a cabo en *Django*, donde se implementó la gestión de usuarios, sin embargo, el equipo decidió cambiar el *framework* de desarrollo y pasar a utilizar *Node.js*.

La razón por la cual se decidió descartar *Django* como *framework* de desarrollo para nuestro proyecto fue la falta de flexibilidad a la hora de implementar dicha gestión de usuarios. *Django* ofrece un modelo de usuario básico cuya modificación (para añadir más atributos, o modificar los ya existentes) resultó complicada, no solo por el desconocimiento del lenguaje, que ningún miembro del equipo había utilizado antes durante la carrera; sino también por la forma en la que *Django* trabaja por dentro.

Node.js

Tras cambiar el *framework* de desarrollo, se volvió a seguir la idea de obtener una gestión básica de usuarios integrada con el prototipo previamente implementado.

Las primeras funcionalidades que se implementaron fueron las de registro, acceso y visualización de la información del usuario en una primera versión del perfil de usuario. En este perfil se añadió un botón de redirección de los usuarios a la pantalla de juego como forma de integración con el prototipo inicial. De este modo, se obtuvo una aplicación a la que los usuarios podían acceder y jugar, aunque la funcionalidad de elegir canción no estaba implementada todavía.

La primera versión de registro de usuarios ofrecía a los mismos un formulario donde introducir sus datos (email, *nickname*, nombre completo y contraseña), así como, de forma opcional, la subida de una foto que se mostraría en su perfil junto con su nombre completo y su *nickname*, como se observa en la figura 4.2. En el caso de que el usuario no proporcionara una foto en el momento del registro, la aplicación le asignaría una foto por defecto.

Figura 4.2: Formulario de registro de usuarios

A medida que avanzaba el desarrollo del proyecto y surgían nuevas ideas a implementar, el módulo de usuarios sufrió varias modificaciones, además de ganar en funcionalidad.

Se introdujo el concepto de avatares en el proyecto. Un avatar es una imagen de un personaje que dota al usuario de ciertas ventajas a la hora de jugar. En concreto, nuestro juego cuenta con seis avatares distintos que ofrecen las siguientes ventajas o poderes:

- Chesire: sumar 500 puntos extra a la puntuación final obtenida por el usuario.
- Blue: multiplicar la puntuación que consigue el usuario por cada pulsación por 1.5.
- Pelusa: salvar la racha de aciertos seguidos en caso de que el usuario falle una pulsación.

- Logan: aumentar el rango de acierto del usuario respecto a la posición de meta.
- Perla: evitar que el usuario sea penalizado por fallar una pulsación.
- Mery: multiplicar por tres la puntuación que consigue el usuario con los dangos.

Con la introducción de este nuevo concepto se decidió modificar el formulario de registro, de forma que, en vez de permitir a los usuarios subir una foto de perfil, se les da la opción de elegir uno de los seis avatares mencionados con su poder correspondiente, como se puede apreciar en la figura 4.3.

Figura 4.3: Formulario de registro con elección de avatar

En la figura 4.4 se muestra el perfil de usuario junto con varios botones; uno que permite al usuario acceder al juego, y otro que permite modificar el avatar y el poder del jugador invitado, funcionalidad que se explicará en la sección 4.2.6: Modo multijugador.

Figura 4.4: Perfil de usuario

Con el desarrollo del proyecto se añadió una funcionalidad basada en la edición de algunos campos del perfil de usuario, en concreto, el email, el nombre completo y el avatar. En el capítulo 6, *Running Tool*, se explicará más en profundidad cómo acceder y cómo funciona el formulario de edición.

Por último, se añadió el cifrado de las contraseñas de los usuarios. Previamente, las contraseñas se guardaban en claro en la base de datos, de forma que cualquiera que accediese a ella podría hacerse con las credenciales de todos los usuarios. En la figura 4.1 se puede observar un ejemplo de cómo se guardaban los datos de un usuario antes del cifrado de las contraseñas.

```
_id: ObjectId("5eac3a570f72c52428c36e77"),
Name: "Niko Peko",
Nickname: "nik",
Email: "niko@email.com",
Password: "nikoelmejor",
Image: "Blue.png",
ImageMulti: "Perla.png"
```

Listing 4.1: Datos de usuario con la contraseña en claro

Para llevar a cabo este cifrado hicimos uso del módulo *bcrypt* de *Node.js* que permite *hashear* las contraseñas de los usuarios haciendo uso de una determinada sal, así como comparar un código *hash* leído de la base de datos con la contraseña en claro introducida por el usuario.

Con esta modificación, se guarda en la base de datos el código *hash* de la contraseña introducida por cada usuario, como se puede observar en la figura 4.2.

```
_id: ObjectId("5eac3a570f72c52428c36e77"),
Name: "Niko Peko",
Nickname: "nik",
Email: "niko@email.com",
Password: "$2b$12$TnjmJgUSNuPIL0CANw1CAuSswxbT3NWJN97YUIKvW9MkjMlnwizV.",
Image: "Blue.png",
ImageMulti: "Perla.png"
```

Listing 4.2: Datos de usuario con la contraseña *hasheada*

4.2.2. Módulo de canciones

El módulo de canciones es el encargado de mostrar a los usuarios una lista de las canciones disponibles que se pueden jugar, junto con sus niveles de dificultad.

Como se ha explicado previamente en la sección 4.1 de este capítulo, el primer prototipo del juego contaba únicamente con una canción. La secuencia de círculos de esta canción estaba representada mediante una variable global con la estructura de un archivo JSON que contaba con un *array* con los tiempos en los que debía dibujarse cada círculo. Dicha estructura es la que más tarde se almacenaría en la base de datos para representar una canción.

La primera funcionalidad implementada en este módulo fue la selección, por parte de los usuarios, de la canción que desean jugar y su carga en la pantalla de juego de forma dinámica. Esta funcionalidad muestra todas las canciones disponibles a los usuarios para que estos elijan cuál desean jugar junto con su nivel de dificultad. Por cada canción se ofrecen tres niveles de dificultad: fácil, medio y difícil. La diferencia principal entre los distintos niveles es la cantidad de círculos que aparecen en la secuencia, tomando como referencia el nivel difícil. De esta manera, el nivel medio cuenta con la mitad, y el fácil con un tercio de los círculos de éste. Se planteó cambiar la velocidad de aparición de los círculos, sin embargo, se tomó la decisión de no aumentar la complejidad del juego para simplificar la experiencia de usuario.

Se añadió un icono de estrella que indica si los niveles asociados a cada canción fueron creados por los administradores del juego o por otros usuarios de la aplicación. Además, se muestra información básica de las canciones, como puede ser el título, el artista y una imagen representativa.

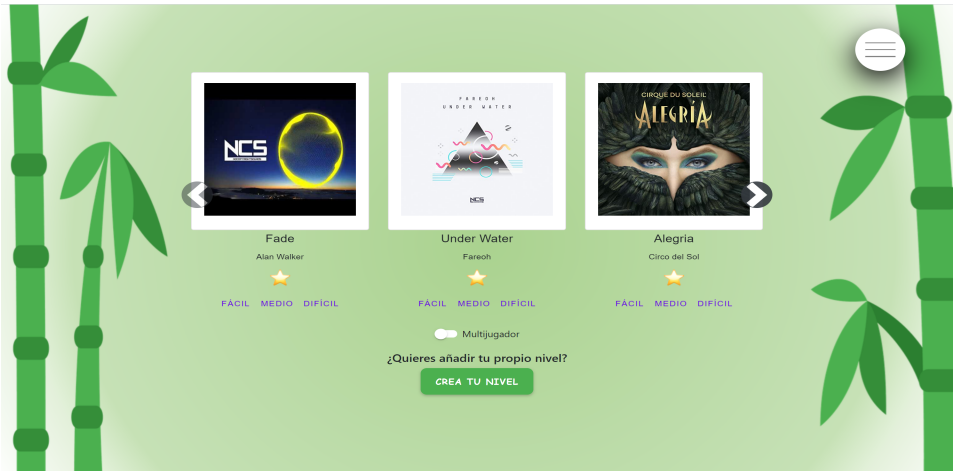

Figura 4.5: Pantalla de selección de canción

Como se puede apreciar en la figura 4.5, la pantalla de selección de nivel permite al usuario cambiar al modo multijugador antes de seleccionar la canción a jugar. En la sección 4.2.6 se profundizará en el modo de juego multijugador, centrándose en las principales similitudes y diferencias con el modo un jugador.

Por último, esta pantalla también incluye un acceso al formulario de creación de nivel. Esta funcionalidad será explicada en detalle en la sección 4.2.4.

4.2.3. Módulo de puntuaciones

El módulo de puntuaciones se encarga de mostrar y almacenar en la base de datos las puntuaciones obtenidas por los usuarios en las distintas canciones.

En un primer momento los únicos módulos desarrollados eran el módulo de usuarios y el de canciones. Al acabar de jugar una canción la puntuación conseguida por el usuario no

se guardaba, por lo que fue necesaria la implementación de un módulo que se encargase de ello.

La primera funcionalidad implementada en este módulo fue la visualización y el guardado de la puntuación obtenida por los usuarios en una canción. Dicha puntuación se muestra al finalizar la partida junto con las puntuaciones obtenidas en intentos anteriores en esta canción.

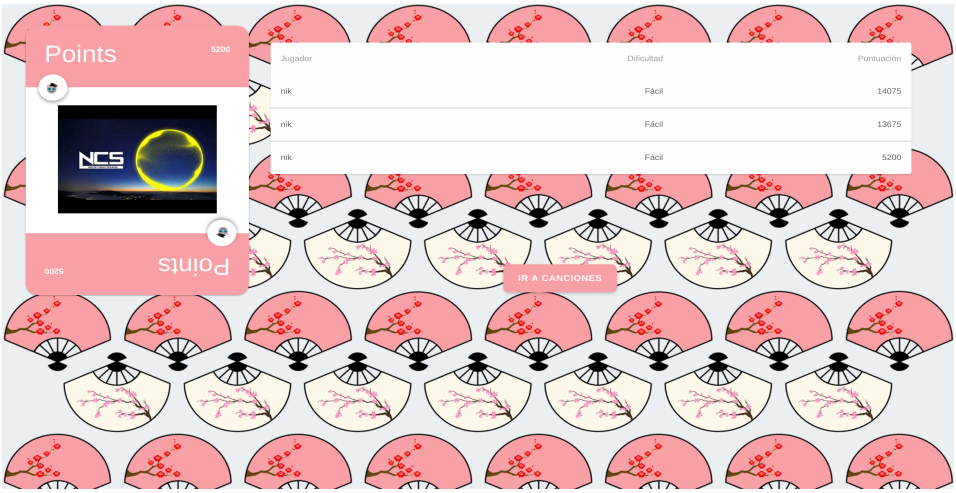

Figura 4.6: Puntuaciones del usuario

Como se puede observar en la figura 4.6, cada fila de la tabla corresponde a un intento del usuario en la canción y contiene la puntuación obtenida además del nivel de dificultad en el que se obtuvo. Una vez visualizadas las distintas puntuaciones se ofrece al usuario la posibilidad de regresar a la selección de canciones.

Además, se implementó la visualización de las puntuaciones globales de las distintas canciones de la aplicación. A diferencia de las puntuaciones mostradas al finalizar una partida, que constan únicamente de aquellas conseguidas por el usuario de la sesión, en esta visualización se incluyen las puntuaciones de todos los usuarios de la aplicación que han jugado cada canción.

Figura 4.7: *Ranking* de puntuaciones

En la figura 4.7 se puede observar la lista de puntuaciones asociadas a la canción «Fade». Para consultar el *ranking* de puntuaciones de otras canciones basta con pasar el cursor por encima de la cancion deseada.

4.2.4. Módulo de creación de nivel

El módulo de creación de nivel permite a los usuarios crear sus propios niveles de juego, que serán accesibles a los demás usuarios de la aplicación.

La idea de ofrecer a los usuarios la posibilidad de crear sus propios niveles surgió en una de las reuniones periódicas con nuestro tutor. Se vió que esta funcionalidad haría el juego más dinámico y amigable, ya que cada usuario podría crear niveles con canciones de su elección dando diversidad al juego.

Esta funcionalidad consta de dos pasos: un primero con un formulario que permite al usuario subir un fichero de audio de una canción, junto con su nombre, autor e imagen (véase figura 4.8); y un segundo en el que se crea la secuencia que se asociará a la canción. La secuecia creada por el usuario se corresponderá con el nivel de dificultad difícil, y las secuencias del resto de niveles se generarán automáticamente.

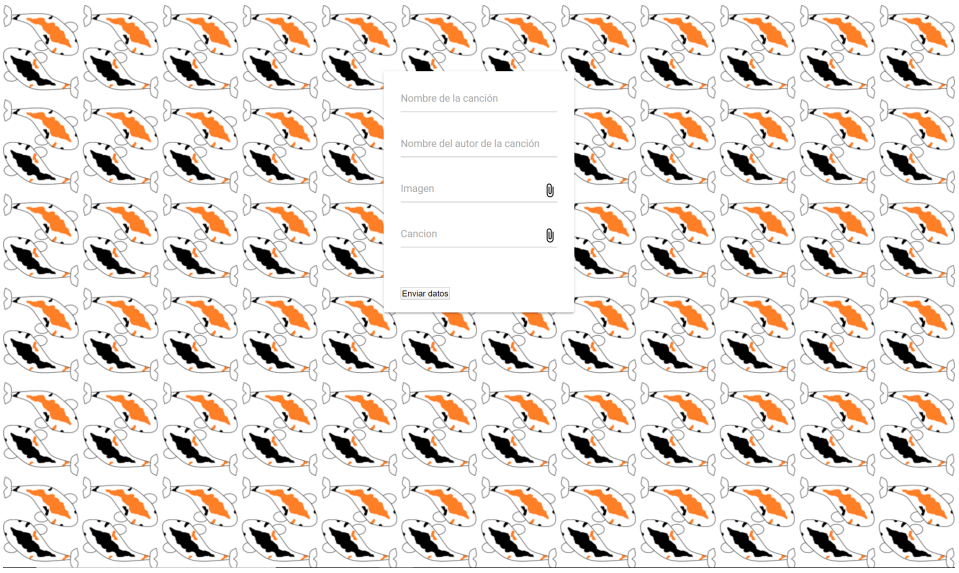

Figura 4.8: Primera fase de creación

Al igual que los módulos expuestos anteriormente, este módulo sufrió varias modificaciones debido a los cambios de perspectiva que se fueron planteando durante su implementación y desarrollo. En un principio, se estudió la posibilidad de almacenar en la base de datos el fichero de audio y la imagen subidos por el usuario en el formulario, sin embargo, esta idea fue descartada debido a la ineficiencia de la transacción.

La implementación final, impulsada por la búsqueda de la eficiencia dentro del juego, se basa en almacenar los ficheros de audio en el servidor y el nombre de éstos en la base de datos con el fin de poder ser recuperados de manera sencilla y eficaz.

Una vez completado y enviado el formulario, los datos introducidos en el mismo se almacenan en la base de datos, mientras que los ficheros se guardan agrupados por tipo, en distintas carpertas dentro del servidor. Estas carpetas almacenan distintos tipos de ficheros: audio, imagen, CSS, *JavaScript*, etc., facilitando a la base de datos la localización y recuperación de los distintos archivos que representan una canción.

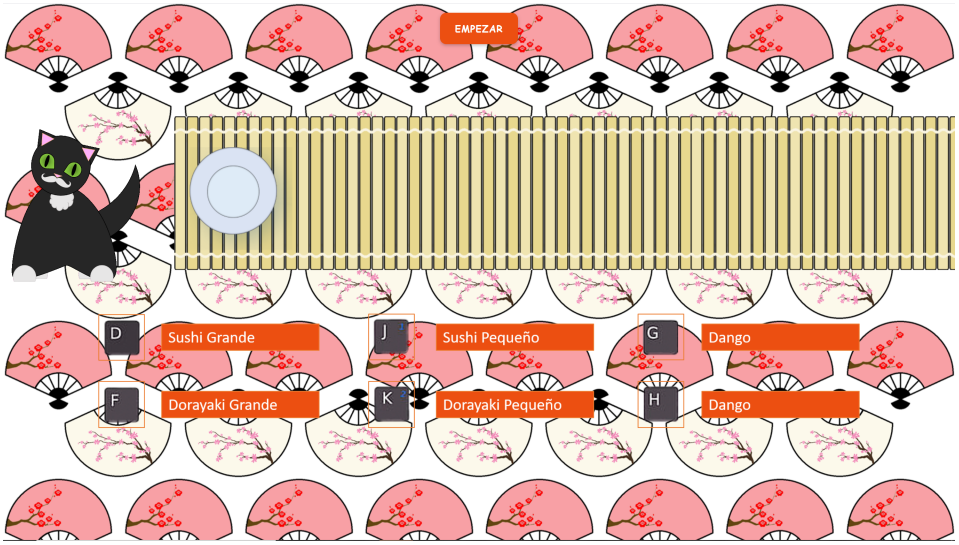

Figura 4.9: Segunda fase de creación

El proceso de creación de la secuencia tiene lugar en la pantalla que muestra la figura 4.9. En esta figura se observa el tablero de juego junto con los distintos tipos de círculos que se pueden añadir y las teclas que se deben pulsar para añadir cada uno. Cabe mencionar que, una vez iniciado, el proceso de creación no se podrá detener hasta que finalice la reproducción de la canción. La secuencia creada no podrá editarse ni durante la creación ni a posteriori.

Los tipos de círculos que se pueden añadir son los expuestos en la sección 1.1 del capítulo de Introducción. Con el fin de añadir dinamismo y versatilidad al juego y a la experiencia de usuario se implementaron para los tipos *sushi* y *dorayaki* dos tamaños, grande y pequeño. Los círculos grandes implican la pulsación simultánea de dos teclas mientras que los círculos pequeños implican una pulsación simple.

Una vez la canción termine de reproducirse, el proceso de creación habrá finalizado; a partir de la secuencia creada por el usuario se generarán de forma automática las secuencias de los distintos niveles de dificultad y se podrán almacenar en la base de datos.

La generación de los niveles de dificultad medio y fácil en el modo un jugador parte de la secuencia creada por el usuario. Para la creación de la secuencia de dificultad media se descarta uno de cada dos círculos introducidos, mientras que para la de dificultad fácil se descarta uno de cada tres círculos. Una vez creadas las tres secuencias correspondientes al modo un jugador, se lleva a cabo la generación de las secuencias del modo multijugador. Esta generación consiste en clonar y modificar las secuencias ya creadas, cambiando el código de las teclas a pulsar para incluir las configuradas para el segundo jugador.

Al finalizar el proceso de generación, conseguimos seis secuencias distintas que equivalen a las tres dificultades del modo un jugador y las tres dificultades del modo multijugador.

Una vez almacenadas todas las secuencias en la base de datos, la nueva canción aparecerá en la pantalla de selección de canción, lista para ser jugada.

Algoritmo de captura de pulsaciones

En esta sección se va a explicar el funcionamiento del algoritmo encargado de capturar la pulsaciones del usuario en el proceso de creación de nivel.

Este algoritmo se encarga, en primer lugar, de comprobar que las teclas pulsadas por el usuario son aquellas que equivalen a los distintos tipos de círculos que pueden añadirse. En segundo lugar, captura el tiempo de cada pulsación, el cual se añadirá a la estructura JSON que representa la secuencia.

Cada pulsación válida registrada equivale a un círculo de la secuencia, que se representa mediante un objeto que contiene el tiempo en milisegundos en el que se dibujará, el tipo de círculo y el código de la tecla o teclas que deberán pulsarse en el juego cuando éste llegue a la posición de meta.

Para indicar al usuario que su pulsación ha sido almacenada en la secuencia con éxito se muestra sobre la posición de meta la imagen correspondiente al tipo de círculo añadido. Debido a que los dangos se corresponden con varias pulsaciones, cuando se añada este tipo de círculo a la secuencia el proceso de creación se bloqueará durante 5 segundos, para evitar que se añadan otros círculos que puedan solaparse con este.

Este algoritmo presenta una pequeña dificultad: en la creación de la secuencia, el tiempo capturado por cada pulsación equivale al tiempo en el que el usuario debería pulsar la tecla mientras juega, sin embargo, los tiempos que se almacenan indican el momento en el que se dibuja cada círculo. Debido a esto, si no se modificasen los tiempos de captura, las secuencias creadas por los usuarios se encontrarían en desfase con la reproducción de la canción. Como solución, se calculó el tiempo que tardan los círculos en llegar a la posición de meta y se restó al tiempo capturado de cada pulsación.

4.2.5. Algoritmo principal del juego

En esta sección se va a explicar el funcionamiento del algoritmo principal de Peko Peko no Neko, encargado de gestionar el flujo del juego. Este algoritmo ha sufrido modificaciones a lo largo del desarrollo del proyecto debido a la implementación de nueva funcionalidad, como por ejemplo, el modo multijugador o la mejora de los avatares con poderes.

El algoritmo se encarga de capturar las pulsaciones del usuario comprobando su validadez dentro del juego, teniendo en cuenta factores extra como los poderes de cada avatar y las rachas de aciertos del usuario.

La captura de las pulsaciones del usuario se realiza a través de la función *onKeyDown* que ofrece *JavaScript*. Esta función nos permite recuperar el código de la tecla pulsada para comprobar si se corresponde con alguna de las del juego. Una vez se ha comprobado que la tecla pulsada es válida se procede a comparar su código con el código de la tecla correspondiente al tipo de círculo que se esté jugando en ese momento.

A la hora de capturar las pulsaciones se ha de tener en cuenta los distintos tipos de círculos implementados en el juego. Los dangos permiten la pulsación de cualquier tecla del

juego, por lo que basta con comprobar la validez de esta. Como se vió en la sección 5.4, los círculos grandes requieren de la pulsación simultánea de dos teclas y, para esta comprobación, se lleva a cabo un marcado de las teclas pulsadas de manera que solo se dará por buena la pulsación simultánea si ambas teclas han sido marcadas.

Una vez se ha comprobado que la tecla pulsada por el usuario es válida y coincide con la correspondiente al círculo que se está jugando, se procede a comparar la posición de dicho círculo con respecto a la posición de meta.

Para obtener la posición del círculo basta con comprobar el valor de la coordenada "x" que se va actualizando a medida que este se desplaza por el canvas. Dependiendo de lo cerca o lejos que se encuentre el círculo con respecto de la meta en el momento de la pulsación, esta se dará por válida o no. Si se encuentra dentro del rango de acierto aumentará la racha de aciertos del usuario, lo que además hará que este consiga mayor puntuación por cada pulsación. Si por el contrario se encuentra fuera de este rango, la racha se perderá y volverá a cero. Independientemente de la racha de aciertos que lleve acumualada el usuario, este podrá conseguir mayor puntuación cuanto más cerca se encuentre el círculo de la meta en el momento de la pulsación.

Debido a la introducción de avatares con poderes en el juego, las comprobaciones mencionadas anteriormente tuvieron que modificarse para llevar a cabo unas acciones u otras en base al avatar escogido por el usuario. Si el usuario eligió el poder que evita que sea penalizado por fallo, cuando una pulsación no sea válida no se verá afectada su puntuación, pero perderá la racha de aciertos acumulada. Si por el contrario se eligió el poder que salva la racha de aciertos, cuando la pulsación no sea válida la racha se mantendrá intacta. Existe también un poder que aumenta el rango de acierto, por lo que la comparación pertinente se lleva a cabo sobre valores superiores.

El algoritmo también tiene en cuenta el modo de juego en el que se esté jugando. Si se seleccionó el modo multijugador, los códigos de las teclas a comprobar varían ya que se ha de diferenciar entre las pulsaciones de los distintos jugadores. El flujo es muy similiar al que sigue el modo un jugador, pero diferenciando en todo momento las pulsaciones de ambos jugadores.

Tras la ejecución de las comprobaciones mencionadas anteriormente se llevan a cabo las modificaciones pertinentes de la parte visual. En caso de acierto o error se dibujan sobre la posición de meta platos de color verde o rojo respectivamente. Además, como se puede observar en la figura 4.10, en función de la racha de aciertos del usuario, se añaden distintas componentes visuales bastante llamativas.

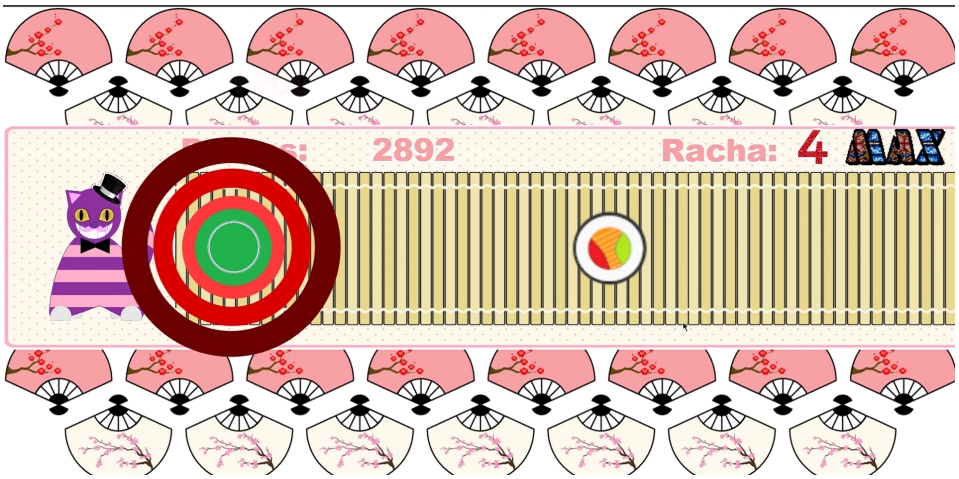

Figura 4.10: Componentes visuales de rachas

En base al incremento o la disminución de la puntuación de los jugadores, se muestra la animación de la suma o la resta de puntos que se puede apreciar en la figura 4.11.

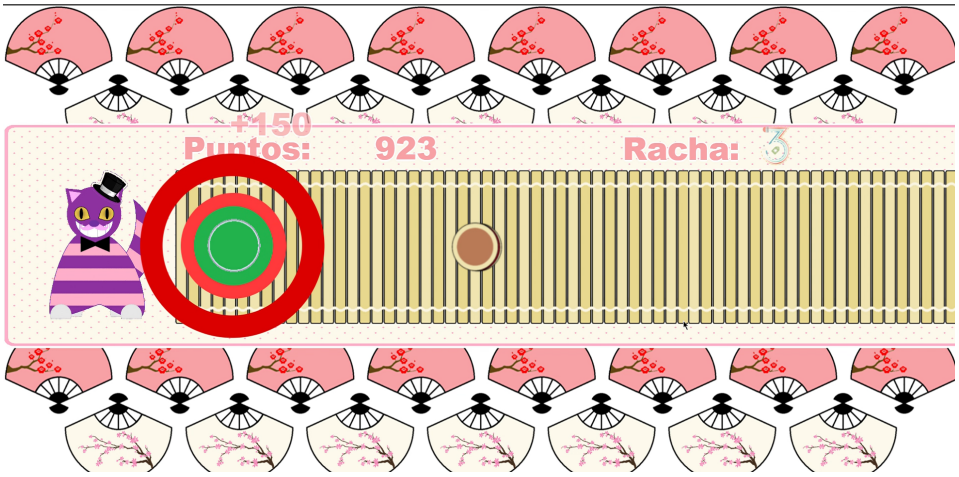

Figura 4.11: Suma de puntos

Al finalizar el nivel y por tanto la ejecución del algoritmo, se almacena toda la información relevante de la partida (identificador de la canción, identificador del usuario, puntuación obtenida, etc.) en la base de datos.

4.2.6. Modo multijugador

En esta sección se explican todos los detalles relativos al juego en modo multijugador. Este modo de juego permite a dos jugadores, el usuario de la sesión y un jugador invitado, jugar desde un mismo terminal sobre la misma secuencia de círculos.

Para permitir que dos jugadores jugasen desde un mismo terminal se implementaron varios cambios en la interfaz, visibles en la pantalla de juego principal.

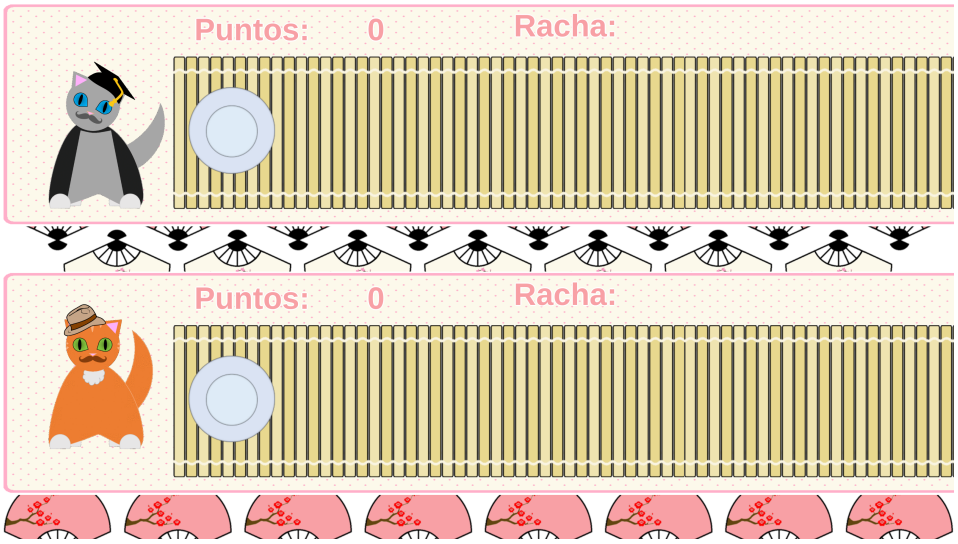

Figura 4.12: Pantalla de multijugador

La mecánica de juego en modo multijugador es exactamente igual a la explicada para el modo un jugador. Como se puede apreciar en la figura 4.12, se duplicó la pista de juego donde se dibujan los círculos, junto con todos sus elementos como la meta, los puntos, la racha, etc., para que cada jugador contase con la suya. De este modo ningún aspecto relativo a un jugador converge con los del jugador contrario.

Para diferenciar a quien pertenece cada pista de juego, se dibuja en cada una el avatar del jugador correspondiente. El avatar asignado por defecto al jugador invitado es el mismo elegido por el jugador de la sesión en el momento del registro, sin embargo, desde el perfil de usuario visto en la sección 4.2.1, se implementó la posibilidad de modificar el avatar del jugador invitado.

Debido a que ambos jugadores comparten tanto pantalla como teclado, las teclas deben repartirse entre ambos de forma que no solapen. Se implementó una generación automática de secuencias de modo multijugador que, a partir de las ya existentes para el modo un jugador o de las creadas por los usuarios, modificaba los códigos de las teclas para incluir las de ambos jugadores.

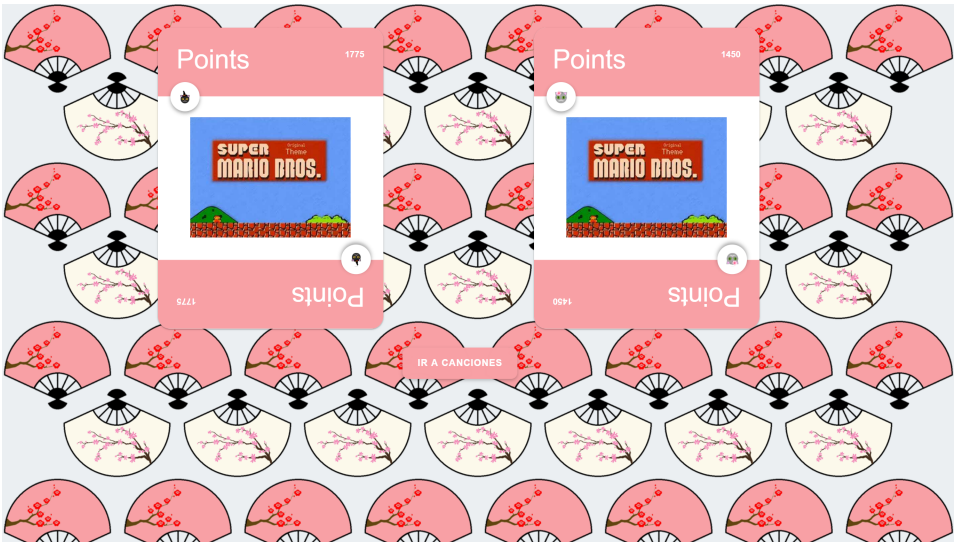

Figura 4.13: Puntuaciones de modo multijugador

Al finalizar el nivel se muestra la puntuación obtenida por cada jugador, como puede observarse en la figura 4.13. Sin embargo, solo se almacena en la base de datos la puntuación del jugador de la sesión, debido a que el jugador invitado no tiene una cuenta asociada en la aplicación.

Este modo de juego presenta varias restricciones. La primera y más importante es el hecho de jugar sobre un mismo terminal, lo que exige a los jugadores compartir teclado. Otra restricción del modo multijugador, como se ha mencionado anteriormente, es que la puntuación obtenida por el jugador invitado no se guarda en la base de datos, por tanto, la única puntuación almacenada es aquella conseguida por el usuario de la sesión.

Capítulo 5

Bases de datos

Como se explica en la sección 3.1, se utilizó como base de datos *MongoDB*, mientras que para facilitar la inserción y modificación de los datos, se empleó la interfaz gráfica de usuario *MongoDB Compass*.

En concreto, nuestra base de datos se llama "PekoPekoNoNeko" y cuenta con cinco colecciones: *Users* y *sessions*, *Songs* y Secuencias, y *Scores*. En la figura 5.1 se puede observar las relaciones entre las distintas colecciones de la base de datos.

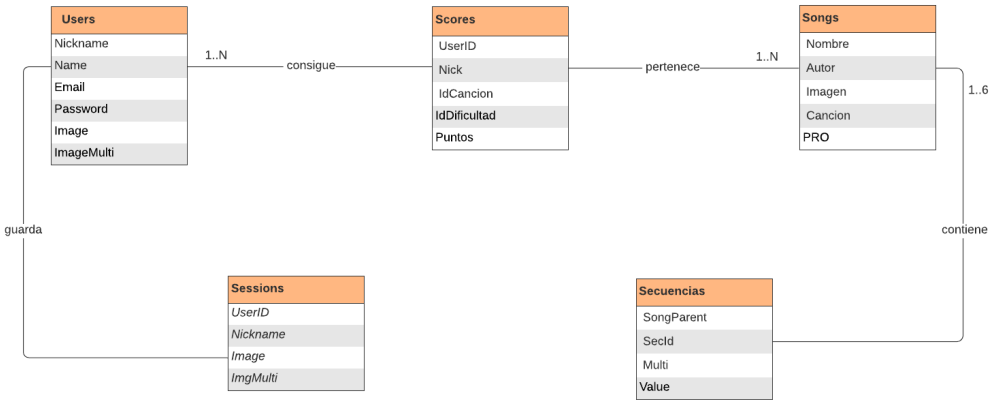

Figura 5.1: Diagrama entidad-relación

A continuación, se contará con más detalle la estructura de los documentos de cada colección, así como las funciones que se implementaron para la manipulación de los mismos.

5.1. Módulo de usuarios

El módulo de usuarios trabaja con dos colecciones de la base de datos: *Users* y *sessions*. La primera guarda la información básica de un usuario de la aplicación, mientras que la segunda guarda la información del usuario de la sesión.

5.1.1. Colección *Users*

La colección *Users* guarda todos los datos básicos de los usuarios, que son los siguientes:

- *Name*: nombre completo del usuario que se mostrará en su perfil.
- *Nickname*: apodo del usuario que lo identifica en la aplicación. Debe ser único para cada usuario.
- *Email*: dirección de correo electrónico del usuario.
- *Password*: contraseña del usuario, cifrada.
- *Image*: nombre del avatar que ha seleccionado el usuario.
- *ImageMulti*: nombre del avatar para el jugador invitado en el caso de jugar en modo multijugador.

En la figura 5.2 se puede ver un ejemplo de un documento JSON de un usuario.

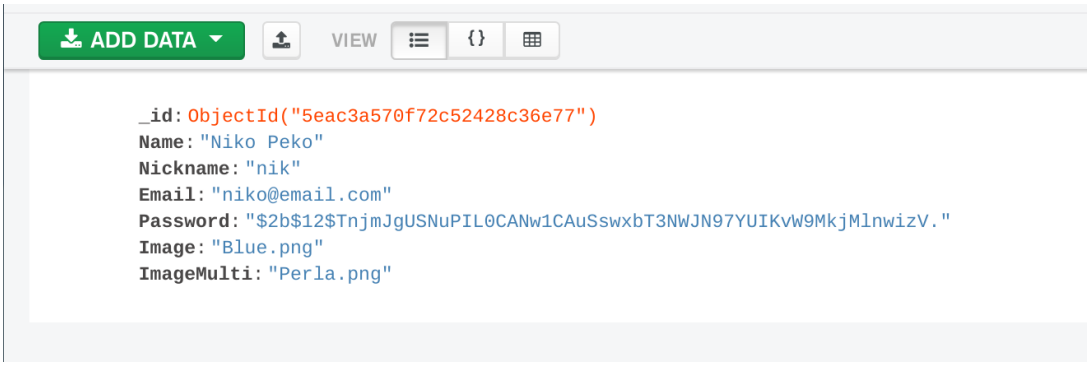

Figura 5.2: Documento JSON que representa a un usuario en la aplicación

Se han implementado las siguientes funcionalidades para manipular los documentos de los usuarios:

- Insertar un usuario en la base de datos.
- Leer un usuario de la base de datos a partir de su identificador.
- Leer un usuario de la base de datos a partir de su *Nickname*.
- Comprobar que no existe otro usuario en la base de datos con el mismo *Nickname* que se ha introducido. Esta función nos permite evitar que se den de alta dos usuarios con el mismo apodo y así garantizar que los apodos son únicos.
- Actualizar los datos del usuario. Como se explica en la sección 3.2.1, un usuario solo puede modificar su nombre, su *email* y su avatar.
- Actualizar el avatar del jugador invitado.

5.1.2. Colección *sessions*

La colección *sessions* guarda en la base de datos los datos del usuario de la sesión. En concreto guarda el identificador del usuario, su apodo, su imagen y la imagen del jugador invitado.

Los documentos de la colección *sessions* son manipulados por el módulo de Express.js "express-session", se crean cuando el usuario accede a la aplicación con sus credenciales y se eliminan cuando el usuario cierra sesión.

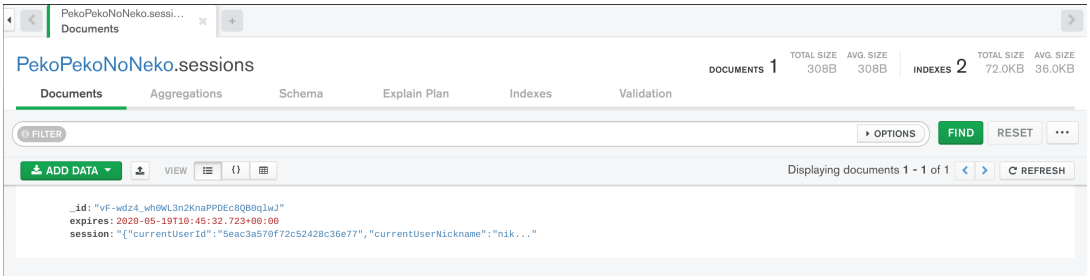

Figura 5.3: Documento JSON que guarda la información del usuario de la sesión

5.2. Módulo de canciones

El módulo de canciones trabaja, al igual que el módulo de usuarios, con dos colecciones: la colección *Songs* y la colección Secuencias. La colección *Songs* guarda la información básica de una canción en la base de datos, mientras que la colección Secuencias guarda, para cada canción, las distintas secuencias de círculos.

5.2.1. Colección *Songs*

Esta colección almacena en la base de datos la infomación básica que representa a una canción, como es su título, autor y una imagen representativa. También almacena el nombre del fichero de audio asociado a la canción, el cuál se utilizará para recuperar el mismo del servidor al comenzar una partida.

En la figura 5.4 se pueden ver documentos que representan canciones dentro de la base de datos.

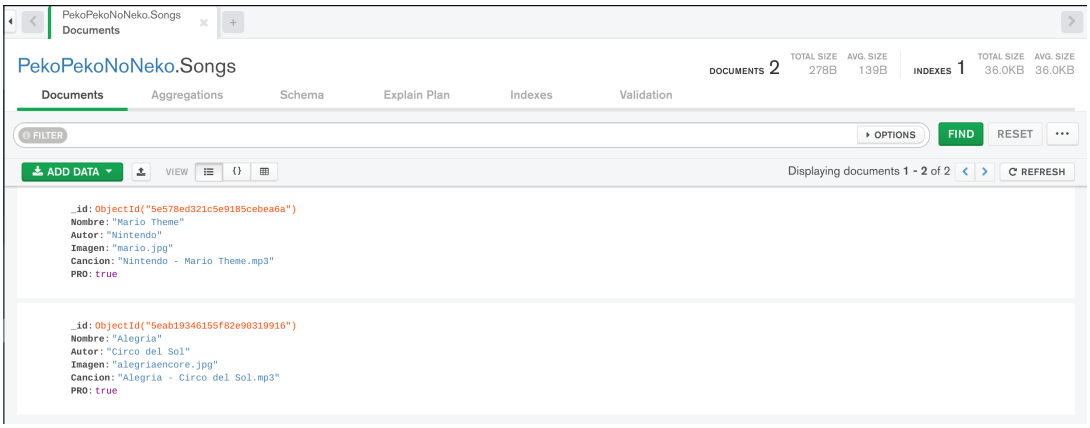

Figura 5.4: Documentos JSON que guardan la información básica de una canción

Para la manipulación de las canciones se implementaron las siguientes funcionalidades:

- Insertar una canción en la base de datos.
- Leer una canción y su secuencia de círculos a partir de su identificador, la dificultad del nivel y el modo de juego.
- Obtener una lista de todas las canciones de la base de datos.

5.2.2. Colección Secuencias

Esta colección guarda las secuencias de círculos para cada canción y cada dificultad, con los siguientes datos:

- *Songparent*: el identificador de la canción a la que pertenece la secuencia.
- Secid: identificador de la dificultad del nivel, 1 para fácil, 2 para medio y 3 para difícil.
- Multi: atributo booleano que indica si la secuencia es de modo multijugador o no.
- *Value*: objeto JSON con la infomación de cada círculo de la secuencia. Para cada círculo a dibujar se guarda el tiempo en milisegundos en el que se dibuja en pantalla, los códigos *JavaScript* de las teclas que deben pulsarse para conseguir el acierto, y el tipo de círculo a pintar.

Un ejemplo de documentos de secuencias se puede ver en la figura 5.5.

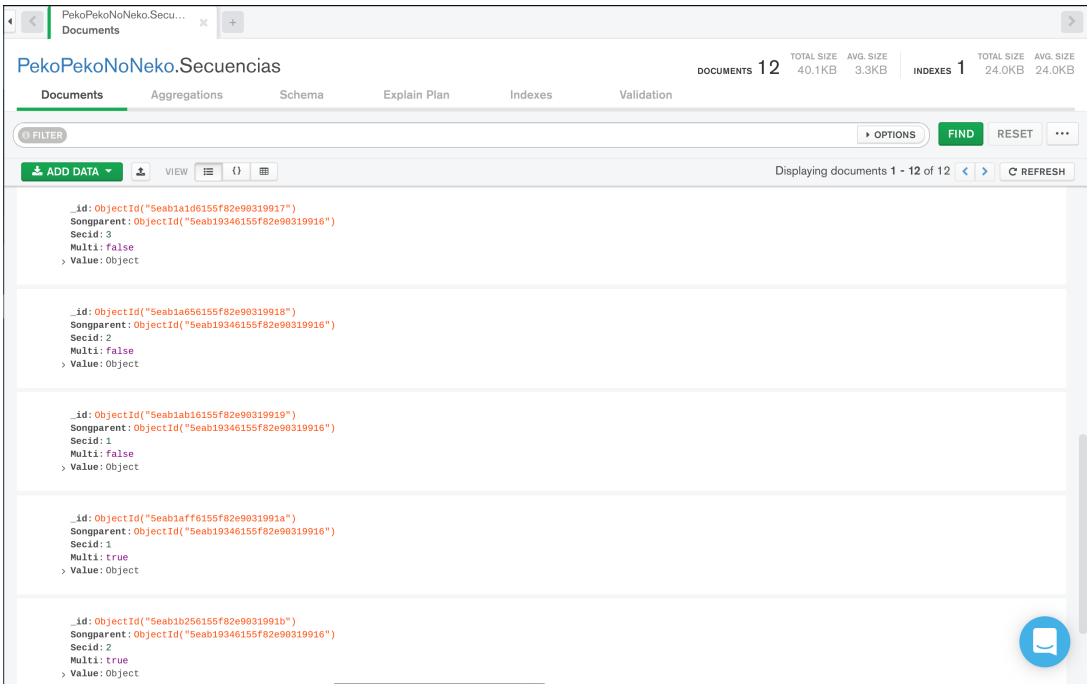

Figura 5.5: Documentos JSON que guardan las secuencias de círculos de distintas canciones

Para la manipulación de los documentos correspondientes a las secuencias se implementó una única función:

– Insertar las secuencias correspondientes a una canción. En concreto se insertan seis secuencias, dos (una en modo multijugador, y otra en modo normal) por cada uno de los tres niveles de dificultad.

5.3. Módulo de puntuaciones

El módulo de puntuaciones trabaja con una única colección llamada *Scores* cuyos documentos asocian una puntuación en una canción y dificultad determinada con un usuario.

5.3.1. Colección *Scores*

Los documentos de la colección *Scores* contienen los siguientes atributos:

- *UserId*: objeto identificador del usuario al que le corresponde la puntuación.
- *Nick*: apodo del usuario al que le corresponde la puntuación.
- IdCanción: objeto identificador de la canción en la que se ha obtenido la puntuación.
- IdDificultad: valor numérico que representa la dificultad en la que se ha obtenido la puntuación. Puede tener tres valores disintos correspondientes a los tres niveles de dificultad.
- Puntos: puntuación que ha conseguido el usuario.

En la figura 5.6 se pueden ver una serie de ejemplos de documentos de puntuaciones.

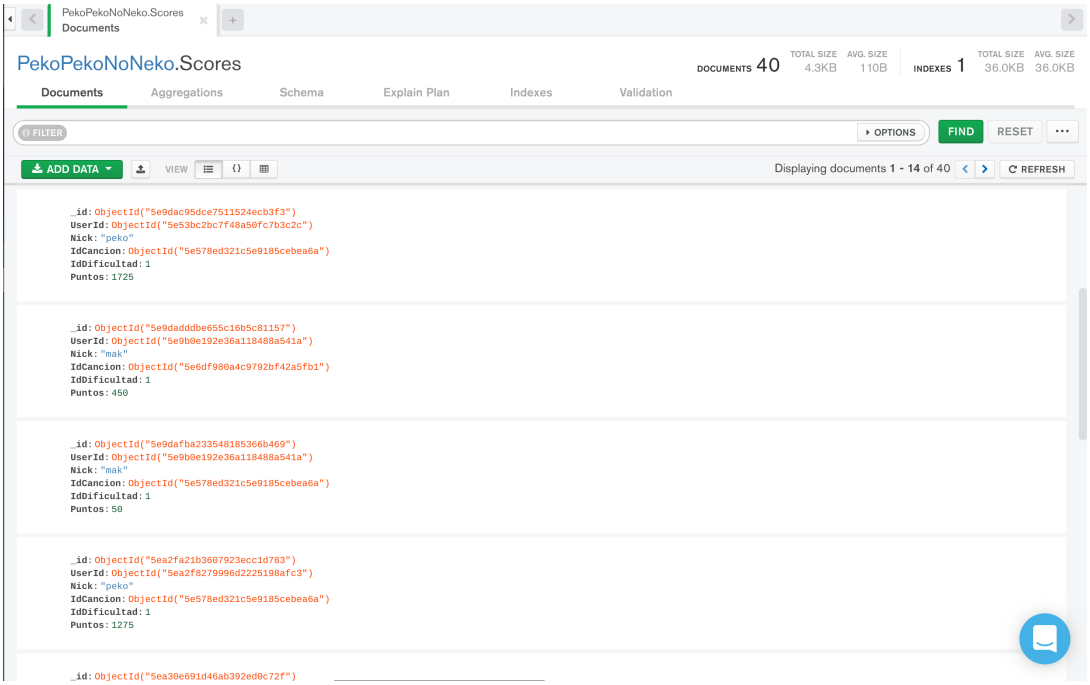

Figura 5.6: Documentos JSON que representan la puntuación de un usuario para una canción

Para la manipulación de las puntuaciones se implementaron las siguientes funcionalidades:

- Insertar una puntuación en la base de datos.
- Obtener una lista de todas las puntuaciones de todas las canciones.
- Obetener una lista de las puntuaciones de un usuario para una canción específica.

5.4. Módulo de creación

El módulo de creación de canciones trabaja con la colección *Songs* y la colección Secuencias que se han descrito en la sección 5.2 de este mismo capítulo.

Desde este módulo se guardan las canciones creadas por los usuarios junto con sus secuencias a través de las funcionalidades descritas en las secciones 5.2.1 y 5.2.2.

Capítulo 6

Running Tool

En este capítulo mostraremos todas las posibles interacciones que puede realizar el usuario con la aplicación. Partiremos de la pantalla de inicio, mostrada en la figura 6.1 y, a partir de esta iremos navegando por todos los módulos implementados.

Para poder avanzar a la pantalla de *Log in* o de registro, basta con presionar cualquier tecla.

Figura 6.1: Pantalla de inicio

6.1. Módulo de usuarios

Como se ha explicado en capítulos anteriores, el módulo de usuarios nos permite registrarnos en la aplicación, inciar sesión y gestionar nuestro perfil de usuario.

6.1.1. Inicio de sesión

La pantalla que vemos en la figura 6.2 es la de inicio de sesión o *log in*. Desde ella podremos acceder a nuestro perfil de usuario introduciendo nuestro *nickname* y contraseña

en los campos correspondientes, y pulsando el botón "*Log in*". También tendremos la posibilidad de registrarnos en el caso de no haberlo hecho ya.

Figura 6.2: Pantalla de incio de sesión o *log in*

Si pulsamos sobre el enlace de "Nuevo usuario", situado en la esquina superior derecha de la pantalla, seremos redirigidos a la pantalla de registro, que puede verse en la figura 6.3.

Figura 6.3: Pantalla de registro

En el formulario de registro deberemos introducir nuestro correo electrónico, una contraseña de al menos 8 caracteres de longitud, nuestro nombre completo, un *nickname* que deberá ser único y nos servirá más adelante para acceder a la aplicación y, por último, elegir uno de los seis avatares disponibles. El proceso de registro no se podrá completar si alguno de los campos del formulario se encuentra vacío.

Una vez hayamos rellenado todos los campos del formulario de registro correctamente, se creará nuestro usuario y seremos redirigidos a la pantalla de perfil.

6.1.2. Perfil de usuario

Una vez hayamos iniciado sesión, o una vez nos hayamos registrado en la aplicación, seremos redirigidos a nuestro perfil de usuario, mostrado en la figura 6.4. Desde aquí podemos llevar a cabo tres acciones diferentes.

Figura 6.4: Pantalla de perfil de usuario

La primera es mostrar el menú desplegable, al cual podemos acceder en la mayoría de las pantallas desde el botón de tipo hamburguesa situado en la esquina superior derecha de la pantalla, como se puede observar en la figura 6.4. Este menú nos ofrece cuatro enlaces distintos, mostrados en la figura 6.5. Mediante el enlace de "Jugar" podremos acceder a la pantalla de selección de canciones, que veremos más a fondo en la sección 6.2.1; pulsando en el enlace de "Puntuaciones" iremos al *ranking* de puntuaciones globales, descrito con más detalle en la sección 6.3; a través del enlace de "Editar perfil" podremos editar algunos de nuestros datos de usuario; y, por último, podremos cerrar nuestra sesión mediante el enlace de "Cerrar sesión".

Figura 6.5: Menú desplegable

Si pulsamos sobre el enlace de "Editar perfil" accederemos a la pantalla mostrada en la figura 6.6. Aquí podremos editar nuestro email y nombre completo, así como el avatar que hayamos elegido al registrarnos. Una vez actualizados los datos seremos redirigidos de nuevo a la pantalla de perfil de usuario.

Figura 6.6: Pantalla de editar perfil

Si por el contrario en el menú pulsamos sobre el enlace de "Cerrar sesión" nos redirigirá a la pantalla de inicio habiéndose cerrado previamente nuestra sesión.

Otra de las acciones que podemos realizar desde la pantalla de perfil de usuario es la de modificar el avatar del jugador invitado, a través del botón de "Cambiar buff jugador 2" mostrado en la figura 6.4. Una vez pulsado dicho botón accederemos a la pantalla que se muestra en la figura 6.7 donde podremos elegir el nuevo avatar que tendrá el jugador

invitado, en el caso de jugar en modo multijugador.

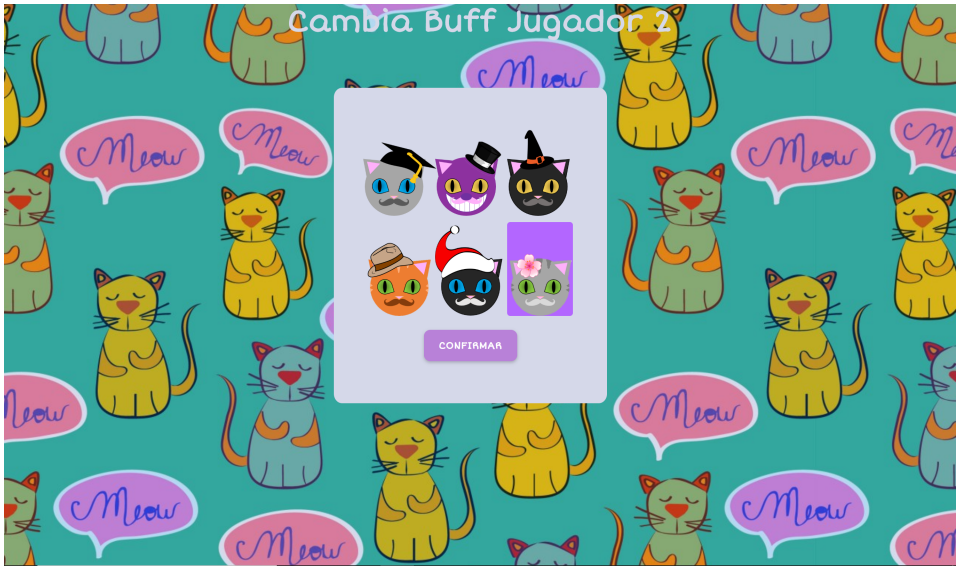

Figura 6.7: Pantalla para cambiar el avatar del jugador invitado

Por último podemos pulsar sobre el botón de “Jugar”, que nos llevará a la pantalla de selección de canciones, donde podremos elegir el nivel a jugar. Esto se explicará con más detalle en la siguiente sección 6.2.

6.2. Módulo de canciones

En esta sección es donde mostraremos el juego en sí, la pantalla de selección de canciones y, posteriormente, la pantalla de juego donde jugaremos ya sea en solitario o en modo multijugador.

6.2.1. Selección de canción

La pantalla de selección de canciones, que se muestra en la figura 6.8, nos permite navegar por las distintas canciones ofrecidas, elegir el nivel de dificultad al cual queremos jugar, acceder al formulario y a la pantalla de creación de nivel, así como desplegar el menú principal.

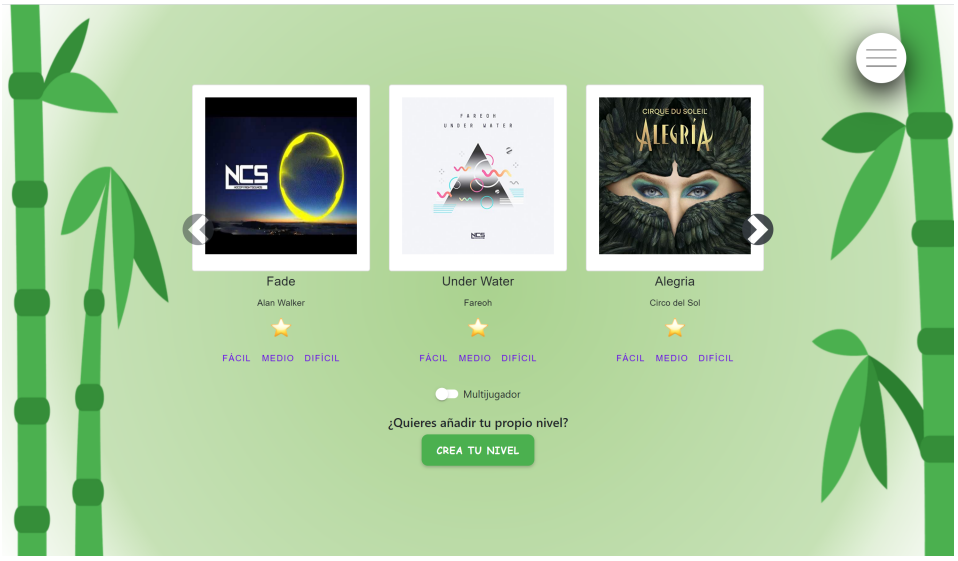

Figura 6.8: Pantalla de selección de canciones

El menú desplegable de esta pantalla es bastante similar al que podemos acceder desde nuestro perfil de usuario. La único diferencia, que podemos observar en la figura 6.9, es que se ha intercambiado el enlace de "Jugar" por el de "Perfil", el cual nos redirige a nuestro perfil de usuario.

Figura 6.9: Menú desplegable desde la pantalla de seleción de canciones

Si pulsamos sobre el botón de "Crea tu nivel" seremos redirigidos al formulario de la primera fase de la creación del nivel, que veremos con más detalle en la sección 6.4.

Mediante las flechas laterales podremos recorrer las distintas canciones ofertadas y, para acceder a la pantalla de juego, descrita en la sección 6.2.2, bastará con que seleccionemos uno de los niveles de dificultad de la canción que hayamos elegido. A estos niveles se accede desde los enlaces "Fácil", "Medio" y "Difícil" situados debajo de cada canción. Además, activando el interruptor ubicado debajo de las canciones, como se observa en la

figura 6.10, podemos acceder al juego en modo multijugador.

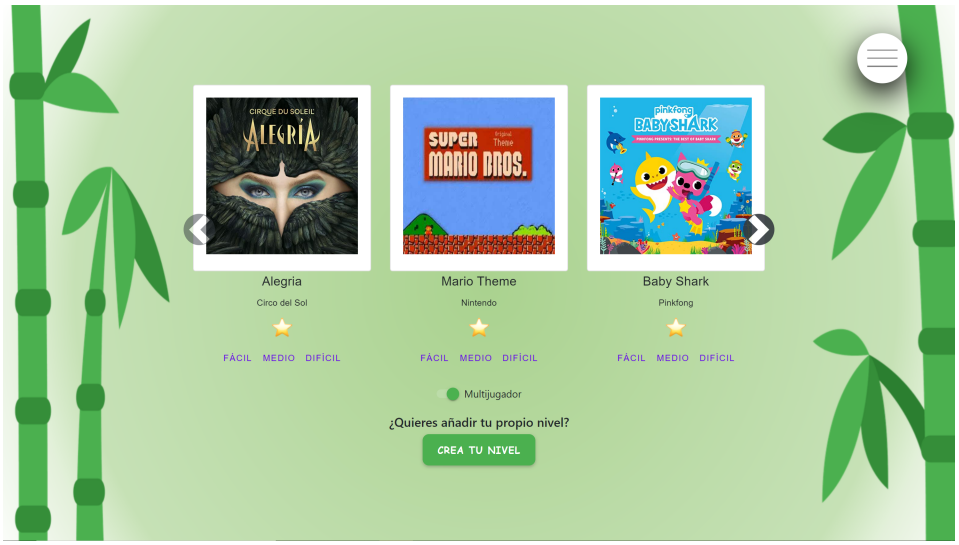

Figura 6.10: Pantalla de selcción de canciones con el interruptor de multijugador activado

6.2.2. Juego

Una vez seleccionado uno de los niveles de dificultad de la canción que queramos jugar seremos redirigidos a la pantalla de juego, que nos muestra la figura 6.11. En esta podemos observar una pista de juego, donde irán apareciendo los distintos círculos que deberemos acertar; el avatar que hayamos elegido, y dos elementos que mostrarán nuestra puntuación actual y la racha de aciertos que llevemos acumulada.

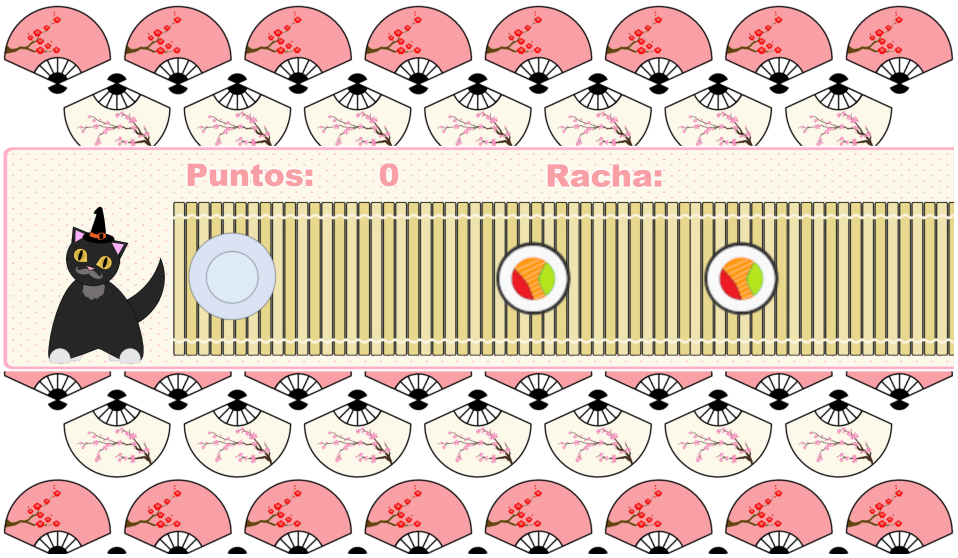

Figura 6.11: Pantalla de juego

Dependiendo del tipo y el tamaño del círculo que aparezca por pantalla deberemos pulsar unas teclas u otras. En general, se utilizan las teclas D, F, J y K para todo el juego, corresponiendo las teclas F y J a los sushis y las teclas D y K a los dorayakis. Cuando

aparezca un círculo pequeño bastará con pulsar cualquiera de las dos teclas correspondientes a su tipo. En cambio, si aparece un círculo grande deberemos pulsar las dos teclas correspondientes a la vez. En el caso de los dangos podremos pulsar cualquiera de las cuatro teclas anteriores tantas veces como queramos mientras se encuentre encima de la posición de meta.

Si queremos jugar en modo multijugador basta con activar el interruptor situado en la pantalla de selección de canciones, tal y como se vió en la figura 6.10. Una vez activado el interruptor y elegida la canción y el nivel de dificultad, la pantalla de juego sufrirá una pequeña transformación con respecto a la pantalla en el modo un jugador, como se puede observar en la figura 6.12.

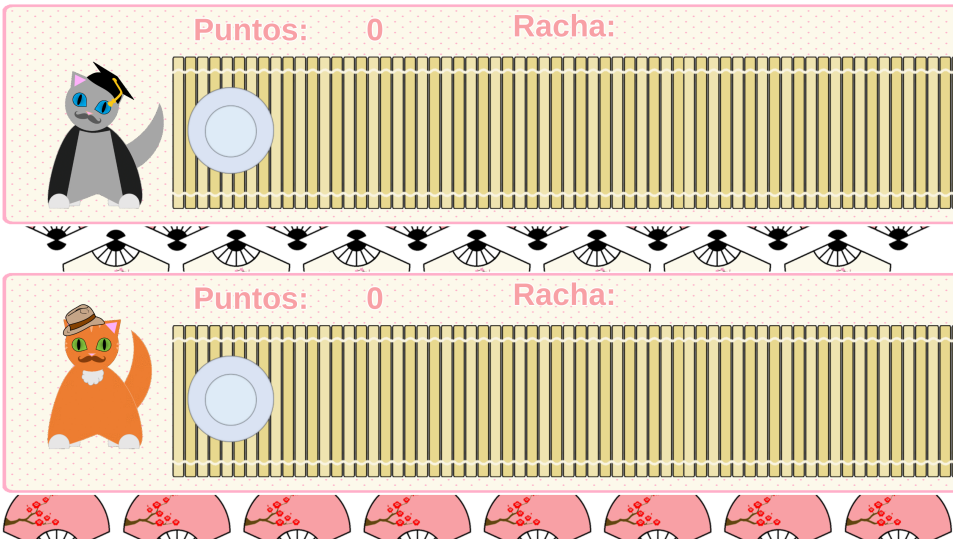

Figura 6.12: Pantalla de juego en modo multijugador

En este modo de juego las teclas que los jugadores deben pulsar cambian. Se utilizan las teclas A, S, D y F para el jugador de la sesión, y las teclas J, K, L y Ñ para el jugador invitado. Cabe destacar que el juego también funciona en aquellos teclados que no cuenten con la tecla Ñ. Por ejemplo, en un teclado inglés se podría jugar pulsando la tecla que se encuentra en su lugar, que en este caso es la de ":" y ";".

Una vez finalizada la canción, tal y como se muestra en la figura 6.13, en nuestra pantalla aparecerá una tarjeta con la puntuación final que hayamos conseguido, y una tabla con las puntuaciones obtenidas en nuestros intentos anteriores en la canción escogida. Para volver a la pantalla de selección de canciones solo deberemos pulsar el botón de "Ir a canciones".

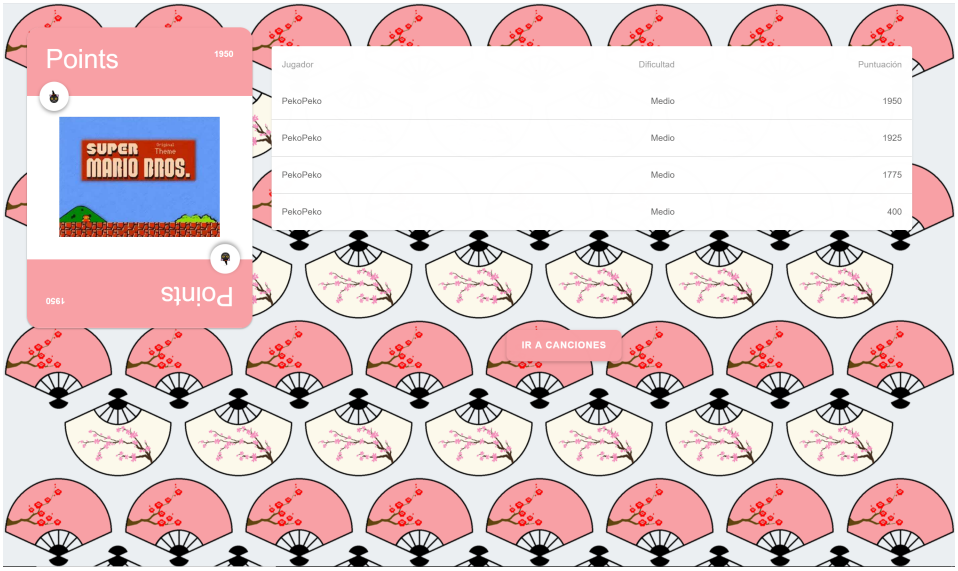

Figura 6.13: Puntuación final conseguida y puntuaciones de intentos anteriores

Si, por el contrario, hemos seleccionado el modo multijugador, en la pantalla solo aparecerán dos tarjetas con la puntuación final de cada uno de los jugadores, como vemos en la figura 6.14.

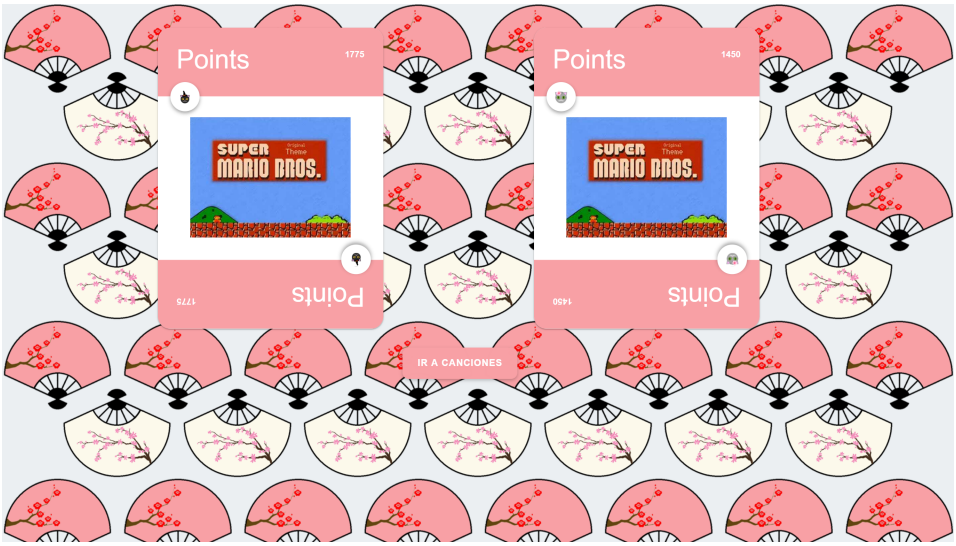

Figura 6.14: Puntuaciones finales de cada jugador en modo multijugador

6.3. Módulo de puntuaciones

Como se ha mencionado en secciones anteriores, podemos acceder a la pantalla de puntuaciones desde el menú desplegable de nuestra aplicación. En ella se muestra una lista con todas las canciones existentes y, cuando el cursor pasa por encima de cada elemento de la lista, se nos desplegará a su vez una tabla con las puntuaciones globales de dicha canción, como podemos observar en la figura 6.15.

Figura 6.15: *Ranking* de puntuaciones

6.4. Módulo de creación de nivel

En este módulo explicaremos cómo podemos crear un nivel de juego propio. Para comenzar la creación de nuestro nivel deberemos pulsar sobre el botón de "Crea tu nivel" que aparece en la pantalla de selección de canciones.

Una vez pulsado dicho botón seremos redirigidos al formulario de la primera fase de la creación de nivel, que puede verse en la figura 6.16. En este formulario deberemos introducir el nombre y autor de la canción, un fichero de imagen que la represente, y el fichero de audio de la misma. Una vez hayamos rellenado todos los campos del formulario correctamente pulsaremos sobre el botón de "Enviar datos" para acceder a la siguiente fase de la creación de nivel.

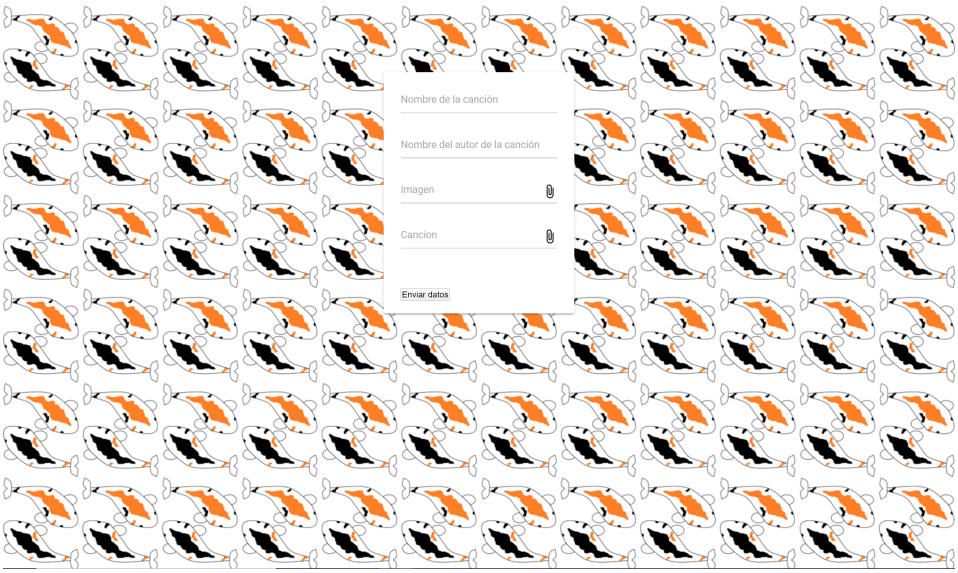

Figura 6.16: Formulario de la primera fase de la creación de un nivel

La segunda fase de la creación de nivel se lleva a cabo en una pantalla muy similar a la de juego, como se puede ver en la figura 6.17. En esta pantalla se nos muestran las distintas teclas que deberemos pulsar para añadir a la secuencia que creemos los distintos tipos de círculos. Para empezar con el proceso de captura de nuestras pulsaciones solo deberemos pulsar el botón de “Empezar”.

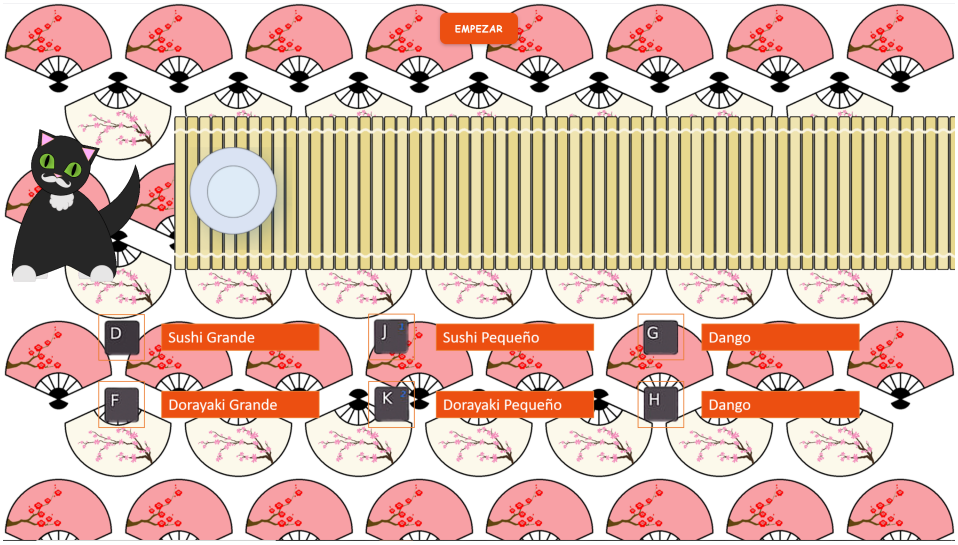

Figura 6.17: Pantalla de creación de nivel

Una vez finalizada la reproducción de la canción aparecerá en la pantalla, tal y como se observa en la figura 6.18, un botón de “Descargar” que guardará la canción que hayamos creado en la base de datos.

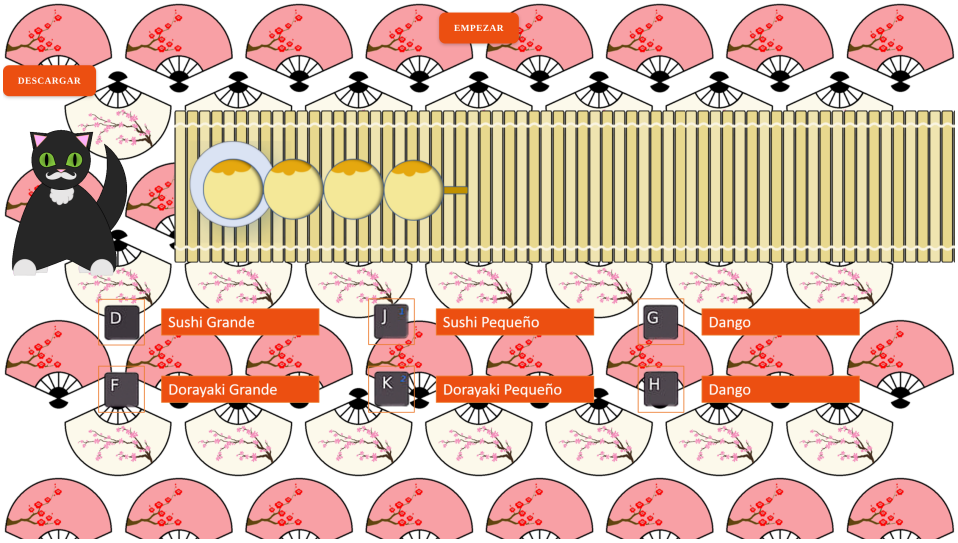

Figura 6.18:

Al pulsar ese botón seremos redirigidos a la pantalla de selección de canciones donde aparecerá la nueva canción que hemos creado junto con sus niveles de dificultad.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

En este Trabajo de Fin de Grado proponemos la implementación de una versión fan o fangame del famoso juego basado en ritmos musicales Taiko no Tatsujin. Nuestra versión se llama Peko Peko no Neko y cambia por completo el diseño que caracteriza a este famoso pasatiempo, transformando las notas musicales en platos tradicionales japonenses y el tambor o taiko en un gato hambriento. El proyecto ha sido implementado utilizando JavaScript como lenguaje principal junto con tecnologías web modernas como es Node.js.

Durante el desarrollo de este trabajo hemos investigado el funcionamiento de distintos frameworks de desarrollo web para poder tomar una decisión valorada y elegir aquel que mejor se adaptase a las necesidades de nuestra aplicación.

Finalizado el proceso de investigación, y una vez quedaron claras las tecnologías que se emplearían, dio comienzo el proceso de implementación. Este proceso se inició con la implementación de un primer prototipo que contaba con las funcionalidades básicas del juego. Para lograr dicho prototipo se llevó a cabo una investigación para determinar la mejor manera de dibujar las notas en la pantalla y de capturar las pulsaciones de los usuarios.

Una vez finalizado dicho prototipo, comenzó el desarrollo por módulos de la aplicación web. El proyecto está dividido en cuatro módulos distintos: usuarios, canciones, puntuaciones y creación de nivel. Cada uno de estos módulos agrupa las distintas funcionalidades que se corresponden con cada uno de los elementos que interactúan en la aplicación. Cuando se obtuvo una aplicación en la que los usuarios podían jugar de forma individual se procedió a la modificación tanto de interfaces como de lógica de juego para ofrecer el juego en modo multijugador. La última funcionalidad implementada fue la posibilidad de los usuarios de crear sus propios niveles de juego.

El balance final del desarrollo de Peko Peko no Neko es positivo. Se ha cumplido con todos los objetivos establecidos al principio de esta memoria y se ha conseguido una aplicación que no solo ofrece la posibilidad de jugar a niveles tanto en modo multijugador como en un jugador, sino, además, la posibilidad de crear niveles de juego propios.

7.2. Trabajo futuro

En esta sección se van a enumerar las posibles mejoras que pueden aplicarse al proyecto en un futuro:

- **Control de pulsaciones a través del dispositivo móvil.** En lugar de capturar las pulsaciones del teclado del usuario, podría implementarse un *driver* que permitiese a este último utilzar su dispositivo móvil como tambor o *"taiko"*. De este modo, en la pantalla del móvil se mostraría la superficie de un tambor con cuatro zonas de pulsación distintas: dos en el centro del tambor, y otras dos en el borde del mismo. Dependiendo del tipo de círculo que aparezca por pantalla el usuario deberá pulsar en una zona y otra del tambor.

- **Implementar el modo multijugador desde dos terminales distintos.** La versión actual del juego permite a los usuarios jugar en modo multijugador en un mismo terminal, compartiendo el teclado. Para mejorar la experiencia de usuario se podría implementar mediante el uso de *web sockets* la posibilidad de jugar en este mismo modo en dos terminales distintos. Además, de esta forma, ambos jugadores estarían registrados en la aplicación y podrían almacenarse en la base de datos las puntuaciones conseguidas por ambos.

- **Secuencias complementarias en el modo multijugador.** Se podría ofrecer a los usuarios la posibilidad de, en modo multijugador, en vez de jugar sobre la misma secuencia, cada jugador contase con su propia secuencia de círculos, que sería complementaria a la del oponente. Se podría, por ejemplo, crear distintas secuencias de círculos que correspondiesen al ritmo de los distintos instrumentos de la canción.

- **Implementación de *lobbies* de juego para contar con niveles multijugador con más de dos jugadores.** La idea de los *lobbies* de juego fue introducida por aquellos juegos *online* con un gran componente multijugador. Se trata de una "sala de espera" donde los jugadores esperan hasta llegar a ser un número determinado para poder comenzar la partida. Esta misma idea se podría implementar en Peko Peko no Neko para contar con niveles en los que puedan participar más de dos jugadores.

- **Permitir al usuario la modificación de la secuencia creada.** Actualmente las secuencias creadas mediante la funcionalidad de "crear nivel" no pueden ser modificadas. Una posible mejora sería la de permitir al usuario, una vez finalizado el proceso de creación, modificar la secuencia de círculos creada.

- **Permitir la elección de dificultad en la creación de nivel.** En la versión actual del juego, la secuencia creada mediante la funcionalidad de "crear nivel" corresponde al nivel "Difícil" de dificultad y, a partir de ésta, se crean los otros dos niveles restantes de dificultad. En este aspecto, podría ofrecerse al usuario la posibilidad de elegir el nivel de dificultad que quiere que corresponda a su secuencia y, a partir de ésta, generar los otros dos niveles restantes.

Capítulo 8

Conclusions and future work

8.1. Conclusions

In this final year dissertation we propose the implementation of a fan version or fangame of the popular game based on musical rhythms *"Taiko no Tatsujin"*. Our version is called Peko Peko no Neko and completely changes the characteristic design of this famous game, swapping the musical notes for traditional Japanese dishes and changing the drum or "taiko" to a hungry cat. This project has been implemented using *JavaScript* as the principal development language along with modern web technologies such as *Node.js*.

During the development of this project we investigated the behaviour of different web development frameworks in order to make a valued decision and choose the one that best suits the needs of our application.

After the investigation process, and once the technologies to be used were clear, the implementation process began. This process began with the implementation of a first prototype that had the basic functionalities of the game. To achieve this prototype, an investigation was carried out to determine the best way to draw the notes on the screen and to capture the keystrokes of the users.

Once this prototype was completed, the modular development of the web application began. The project is divided into four different modules: users, songs, scores and level creation. Each of these modules groups the different functionalities that correspond to each of the elements that interact in the application. Once we obtained a functional application in which users could play individually, both the interfaces and the game logic were modified to offer the game in multiplayer mode. The last functionality implemented was the possibility for users to create their own game levels.

The final assessment of the development of Peko Peko no Neko is positive. All the objectives established at the beginning of this report have been met and an application that not only offers the possibility of playing levels, both in multiplayer and single player mode, but also the possibility of creating and playing self-made game levels has been obtained.

8.2. Future work

This section enumerates possible improvements that can be applied to the project in the future:

- **Tap control through a cell phone device.** Instead of capturing the keystrokes from the user's keyboard, a driver that allows to use their cell phone as a drum or *"taiko"* could be implemented. This way, the cell phone screen would show the surface of a drum with four different tap zones: two in the middle of the drum, and the other two at the edge of it. Depending on the type of circle that appears on the screen, the user must tap one part or another of the drum.
- **Implement multiplayer mode from two different terminals.** The current version of the game allows users to play in multiplayer mode on the same terminal by sharing the keyboard. To improve the user experience, the possibility of playing in this same mode from two different terminals could be implemented through the use of web sockets. In addition, this way both players would be registered in the application and both their scores could be stored in the database.
- **Complementary note sequences in multiplayer mode.** Users could be offered the possibility of, in multiplayer mode, instead of playing the same sequence, each player would have their own sequence of notes, which would be complementary to that of their opponent. For example, different sequences of notes that correspond to the rhythm of the different instruments of the song could be created.
- **Implementation of game lobbies to have multiplayer levels with more than two players.** The idea of game lobbies was introduced by those online games with a large multiplayer component. It is a "waiting room" where players wait until a certain number of participants is reached to start the game. This same idea could be implemented in Peko Peko no Neko to have levels in which more than two players can participate.
- **Allow the user to modify the created sequence.** Currently, sequences created by using the çreate level"functionality cannot be modified. A possible improvement would be to allow the user, once the creation process is finished, to modify the sequence of notes created.
- **Allow the choice of difficulty in level creation.** In the current version of the game, the sequence created by the çreate level"functionality corresponds to the "Difficult"level of difficulty and, from this, the other two remaining levels of difficulty are created. In this aspect, the user could be offered the possibility to choose the level of difficulty that he wants to correspond to his sequence and, from this one, generate the other two remaining levels of difficulty.

Capítulo 9

Contribuciones al proyecto

En este capítulo, cada uno de los integrantes del equipo explicará en detalle sus contribuciones al proyecto, de forma que se pueda apreciar el trabajo realizado desde una perspectiva individual. Cabe destacar que durante el desarrollo de este proyecto se ha realizado un trabajo colaborativo entre todos los miembros del equipo, repartiendo el trabajo a realizar de forma equitativa.

9.1. Alejandro Pedrosa García

Mi trabajo comienza con la asistencia a las primeras reuniones para acordar el proyecto que hemos realizado, decidiendo el alcance que iba a tener, lo que era factible o no, y otorgando al grupo una opinión bastante crítica sobre los aspectos referentes al proyecto.

Tras esas primeras reuniones tocaba empezar a trabajar; lo primero que hice fue buscar algún punto inicial para componer el algoritmo principal y es cuando encontré una especie de emisor de partículas que nos dio un punto de partida. También participé en la toma de decisión del tipo de base de datos y del marco de desarrollo inicial, que fue Django, pero no llegó a buen término. Una vez conseguida la base del proyecto fui realizando tareas y gestionando en la medida de lo posible lo que íbamos haciendo cada uno de los integrantes. Conseguí que se dibujaran los círculos siguiendo una secuencia de tiempos, aunque esa versión aún era bastante rudimentaria.

A continuación, nos centramos en recabar información sobre la captura de las pulsaciones de teclas, ya que no era solo capturar la pulsación sino capturar todas las pulsaciones de todas las teclas y comprobar si eran única y exclusivamente las necesarias para considerarlas como acierto; y conseguimos llevarlo a cabo. Participé en las reuniones, tanto con el tutor como con las compañeras, en las que terminamos decidiendo cambiar el marco de desarrollo y la base de datos a Node y MongoDB. Ayudé a generar la base de datos y gestioné parte de sus consultas, ya fueran las subidas de datos (como es el caso de añadir las distintas secuencias de cada canción) o modificaciones en esta.

Además, en el apartado de juego, también he desarrollado la parte del código que gestiona la gestión de los poderes que otorgan los distintos avatares, la suma y resta de puntos en función del tipo de acierto o fallo, el correcto dibujado de los círculos (que finalmente se han transformado en sushis, dorayakis y dangos), la gestión de rachas, y la correcta

captura de todas las pulsaciones que se hacen en ambos modos de juego.

En otra reunión se decidió implementar el módulo de creación de canciones por parte del usuario, y tuvimos que buscar algo similar a esta idea. No lo encontramos, así que decidimos pensar la idea en que podíamos realizar esto. Desarrollé la primera pantalla de este módulo, la del formulario, de forma que se limitaran los archivos requeridos a imagen en el campo de imagen y a audio en el campo de audio, y se subieran a la base de datos de forma que en la siguiente pantalla se pudieran utilizar. También desarrollé este cambio de pantalla y gran parte del código que compone la segunda fase de creación, ya que yo era el que mejor conocía el algoritmo principal, del que extrajimos la parte de captura de pulsaciones, que era necesaria para crear el nivel. Además fui el que añadió la conversión de la secuencia que iba generando el usuario al crear el nivel a las otras cinco versiones de la secuencia que eran necesarias para componer las dificultades y los dos modos de juego. Además gestioné gran parte de la subida de estas secuencias a la base de datos.

Respecto al apartado de puntuaciones, fui el responsable de desarrollarlo, aunque no el único. Desarrollé la transición de la pantalla de juego a la de puntuaciones y la muestra de la puntuación y de todas las referentes a la canción jugada, trayéndolas de la base de datos.

Otro aspecto que me concierne es la pantalla de selección de canciones, de la que realicé una parte importante del desarrollo, separando el apartado visual.

En lo relativo a la parte gráfica del juego y gracias a la cantidad de CSS que hay en internet, pude desarrollar el carrusel que se muestra en la pantalla de selección, el menú que se muestra en todo el juego, las rachas que se aprecian al conseguirlas en la pantalla de juego, el aspecto visual de la suma de puntos y la animación de la muestra de las puntuaciones al terminar la canción.

Finalmente, aparte de aportar una de las opiniones realistas acerca del tiempo disponible para terminar el proyecto y de lo que podríamos realizar y lo que no en las distintas reuniones (también telemáticas debido a la COVID-19), ayudé en la toma de decisiones de lo referente al complejo visual de la totalidad del juego. Además me gustaría resaltar los problemas que hemos tenido para realizar el proyecto durante los meses de cuarentena, ya que no hemos podido comunicarnos de la mejor forma posible para realizar el proyecto.

9.2. Leila Ruiz Casanova

Mi trabajo comienza con la búsqueda de tutor y por consiguiente de trabajo de fin de grado. Una vez encontré ambas cosas, vimos que era mucho trabajo para una sola persona así que tuve que buscar compañeros. Por suerte encontré a Victoria y a Alejandro, les expliqué la idea y les gustó. Se lo comuniqué al tutor y ya quedaba estar a la espera de que comenzara el curso.

Al principio del curso tuvimos la primera reunión con nuestro tutor. Ahí acordamos que estas reuniones iban a tener una frecuencia cada quince días. En ellas enseñábamos el trabajo que habíamos realizado y decidíamos las tareas y puntos que arreglar para la

próxima reunión. Durante estas reuniones, me encargué de tomar acta de todo lo que se decía en ellas.

Una vez tuvimos nuestra primera reunión, lo primero que hicimos fue familiarizarnos con el proyecto web en el que nos hemos basado. Investigamos cómo era el proyecto, lo que tenía, lo que no y lo que podíamos hacer nosotros. Aquí decidimos las funcionalidades que íbamos a implementar en nuestra aplicación, tanto las que ya existían como las nuestras propias.

Después de esto ya era hora de comenzar con el proyecto. Para poder trabajar todos a la vez, decidimos usar *Github* como herramienta de control de versiones. Me encargué de crear el repositorio donde alojaríamos nuestro proyecto, organizarlo y añadir a mis compañeros cómo colaboradores. Gracias a esta herramienta todos pudimos ir revisando código, arreglando errores y realizando pruebas a medida que avanzábamos con el proyecto.

A partir de aquí comenzamos a organizarnos usando la herramienta de gestión de tareas *Microsoft To-Do*, la cual propuse, organicé y configuré para su uso durante todo el proyecto. También participé activamente en la gestión de proyecto con el propio plan de proyecto sus prioridades y sus estimaciones. A lo largo del proyecto también participé en el diseño en general de toda la aplicación, tanto en las decisiones sobre qué hacer y donde.

Lo primero fue decidir conjuntamente el tipo de la base de datos y el *framework* de desarrollo con el que trabajaríamos, aunque poco después decidimos cambiarlo.

En cuanto al desarrollo del proyecto comencé con la implementación de la lógica de las puntuaciones, calculando los márgenes de acierto y error y su correspondiente puntuación. También ayudé a mis compañeros durante todo el desarrollo en la implementación de nuestra aplicación, tanto en la parte lógica como visual. Además de encargarme de la parte de los distintos avatares y la implementación de la selección de los mismos.

Por último, me encargué principalmente del diseño gráfico de nuestra aplicación, buscando todo el rato simplicidad, consistencia y una buena composición. Tanto con los colores como la temática y la disposición de las cosas. Además tratando de aportar las características necesarias para obtener una buena interfaz de usuario para poder ofrecer una buena experiencia de usuario. También cree la mayor parte de las imágenes y gifs usados en la aplicación estableciendo todos los *assets* gráficos como propios. De esta manera podemos asegurar que la aplicación no va a tener problemas de licencias y podemos liberar todo el código sin ningún problema.

9.3. María Victoria Barylak Alcaraz

Al principio del proyecto me encargué, junto con mis compañeros, de investigar el proyecto web ya desarrollado, mencionado en el capítulo 2, en el que nos hemos basado. El objetivo que buscábamos con esta investigación era el de obtener una idea más clara sobre la estructura que debería tener nuestro proyecto.

Debido a que el proyecto mencionado estaba desarrollado en un lenguaje con el cual niguno de los miembros del equipo estábamos familiarizados, y utilizaba tecnologías y elementos web que desconocíamos, me encargué de redactar un correo electrónico dirigido al propietario del repositorio de *GitHub*. En este correo planteábamos a dicho usuario una serie de dudas sobre el mecanismo que seguía a la hora de dibujar los círculos sobre la pista de juego. Lamentablemente, no obtuvimos respuesta.

Otro de los aspectos en los que participé activamente junto con mis compañeros fueron las reuniones periódicas que llevábamos a cabo con nuestro tutor. En estas reuniones exponíamos el trabajo que se había realizado hasta ese momento, y discutíamos las tareas a realizar para la siguiente reunión.

Al comienzo del desarrollo participé en la investigación del funcionamiento y la posible implementación del proyecto utilizando *Django*, el primer *framework* de desarrollo que se propuso. Implementé un módulo de usuarios básico, con las funcionalidades de registro y acceso. Sin embargo, y como se ha explicado en el capítulo 4, no continuamos el desarrollo con este *framework*.

Debido a las dificultades que encontrábamos durante la implementación en *Django*, propuse en una reunión con el tutor el cambio de *framework* de desarrollo para pasar a utilizar *Node.js*.

Una vez el equipo entero estuvo de acuerdo con este cambio, llevé a cabo la creación del proyecto en *Node.js*, así como la reestructuración del repositorio de *GitHub* para que alojara el nuevo proyecto. También asistí en la migración al mismo de todo aquello que habíamos implementado en *Django*.

Debido a que era el miembro del grupo que más familiarizado estaba con el nuevo *framework* de desarrollo elegido, ayudé a mis compañeros en la transición a este nuevo marco, dando solución a las posibles dudas de desarrollo o conceptuales que les podían surgir durante las primeras semanas. De este modo pudimos aplanar en cierta medida la curva de aprendizaje para que el ritmo de desarrollo no sufriese una ralentización considerable.

En cuanto al desarrollo mismo del proyecto me encargué de la implementación, tanto de la parte visual como de la lógica, del módulo de usuarios. Como se explica en el capítulo 4, este módulo incluye el registro y acceso de los usuarios a la aplicación, la visualización de la información de los mismos en un perfil, y la modificación de los datos.

También implementé la lógica que se encarga del cifrado de las contraseñas de los usuarios.

Una vez finalicé la implementación del módulo de usuarios, asistí en la implementación de algunas funcionalidades del módulo de puntuaciones. En concreto, desarrollé la lógica y la vista encargada de mostrar un *ranking* de puntuaciones globales para cada canción.

También asistí en el desarrollo de la lógica que controla el cambio de modo de juego. Como se explica en el capítulo 6, este cambio se lleva a cabo activando un *toggle* en la pantalla de selección de canciones. Me encargué de conseguir el correcto envío del valor

del modo de juego desde la vista hasta el servidor para su posterior uso.

Por último, y ya como detalles visuales, implementé la aparición de mensajes de error en las vistas, para aquellos casos en los que tuviese lugar algún fallo interno del servidor.

Índice de figuras

Bibliografía

[1] Usuario de *GitHub* Bui. Taiko web. https://taiko.bui.pm/.

[2] Visual studio code. https://code.visualstudio.com/, 05 de mayo de 2020.

[3] Shannon Bradshaw, Eoin Brazil, and Kristina Chodorow. *MongoDB: The Definitive Guide*. O'Reilly Media, 2019.

[4] Basarat Syed. *Beginning Node.js*. Apress, 2014.

[5] Evan M. Hahn. *Express in Action. Writing, building, and testing Node.js applications*. Manning, 2016.

[6] Github. https://github.com/, 05 de mayo de 2020.

PASCAL
ENERO 2018
Ult. actualización 25 de junio de 2020
LaTeX lic. LPPL & powered by TEFLON CC-ZERO